Gráfico com R

Vanderlei Júlio Debastiani

30 de Abril de 2018

Conteúdo

1 Introdução

R é uma linguagem e ambiente de programação estatística e gráfica. Considerando apenas a parte gráfica o R há muitos recursos. No R é possivel fazer de gráficos simples de barras até figuras complexas como mapas e filogenias.

Existe muitos parâmetros gráficos do R. Para fazer um gráfico com qualidade de publicação é preciso usar muitos parâmetros. Para começar a fazer os gráficos não é muito fácil, pois a documentação não é apresentada visualmente.

A proposta deste texto é mostrar um pouco sobre a construção de gráficos. O objetivo não é explorar todos os parâmetros e nem esgotar as possibilidades gráficas do R, apenas mostrar o começo para facilidar a vida daqueles que pretendem usar o R para fazer gráficos.

Os códigos apresentados podem ser executados passo-a-passo para entender o funcionamento de cada argumento.

Os dados de exemplos usados nesta página provem na maioria de conjuntos de dados inclusos em alguns pacotes do R. Para carregar basta usar a função data com o nome do conjunto de dados. Além disso, antes de executar o cógido dos gráficos os dados foram organizados em um dataframes. Isso facilita o entendimento e para padromizar a indezação das variáveis.

2 Funções para gráficos

Para construir um gráfico usa-se função generica plot. A função plot é uma função que produz gráficos de maneira automático dependendo do tipo de informação e classe do objeto passado para ela. Além disso, existem várias funções auxiliares para adicionar elementos ao gráfico.

2.1 Tipos de gráficos

Dados

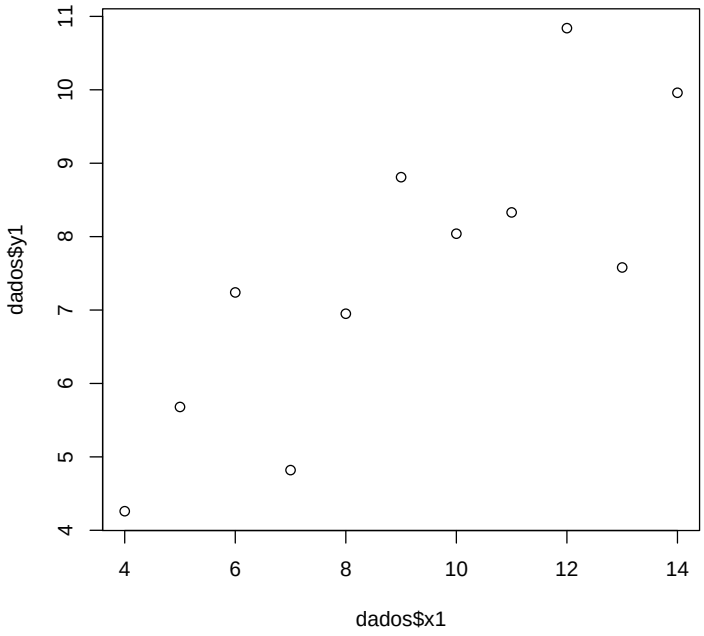
Dados anscombe é um conjunto de dados didáticos para uso em regressoes lineares. São 4 pares de dados com variável independente e dependente totalizando 11 observações.

```
data(anscombe)
dados <- anscombe
str(dados)

## 'data.frame': 11 obs. of  8 variables:
##  $ x1: num  10 8 13 9 11 14 6 4 12 7 ...
##  $ x2: num  10 8 13 9 11 14 6 4 12 7 ...
##  $ x3: num  10 8 13 9 11 14 6 4 12 7 ...
##  $ x4: num  8 8 8 8 8 8 8 19 8 8 ...
##  $ y1: num  8.04 6.95 7.58 8.81 8.33 ...
##  $ y2: num  9.14 8.14 8.74 8.77 9.26 8.1 6.13 3.1 9.13 7.26 ...
##  $ y3: num  7.46 6.77 12.74 7.11 7.81 ...
##  $ y4: num  6.58 5.76 7.71 8.84 8.47 7.04 5.25 12.5 5.56 7.91 ...
```

Gráficos de pontos

Se os x e y são dois vectores numéricos o gráfico resultante é de pontos, sendo mostrado x contra y.

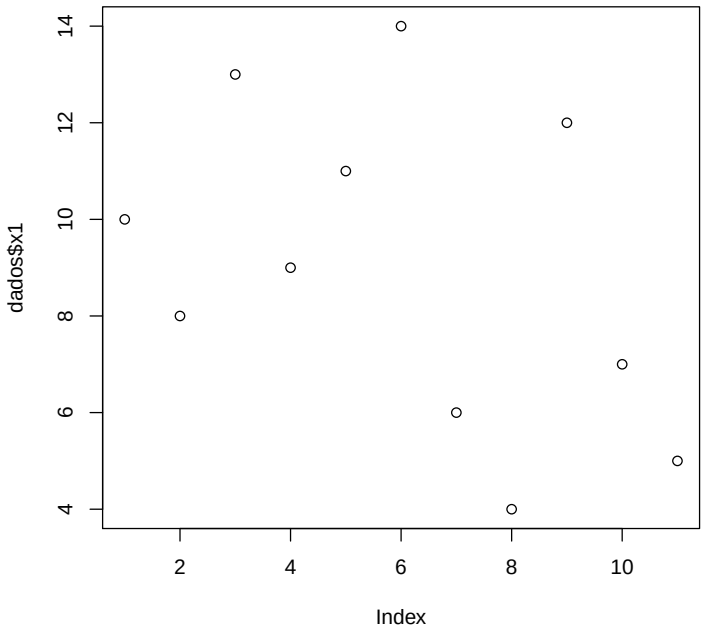
```
plot(dados$x1, dados$y1)
# plot(y1 ~ x1, data=dados) produz o mesmo resultado.
```



Gráficos de séries

Se apenas um vectores numéricos é fornecido a função gera uma série de dados, seguindo a ordem que os valores estão no vector.

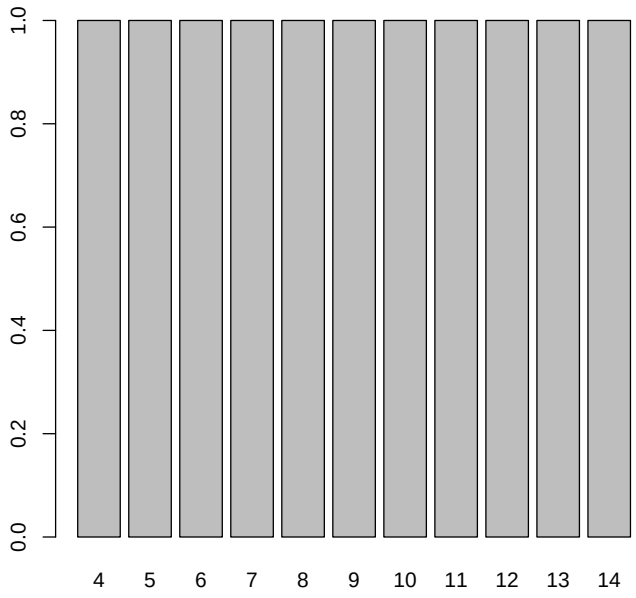
```
plot(dados$x1)
```



Gráficos de barras

Se apenas um vectores do tipo factor é fornecido a função gera gráficos de barra.

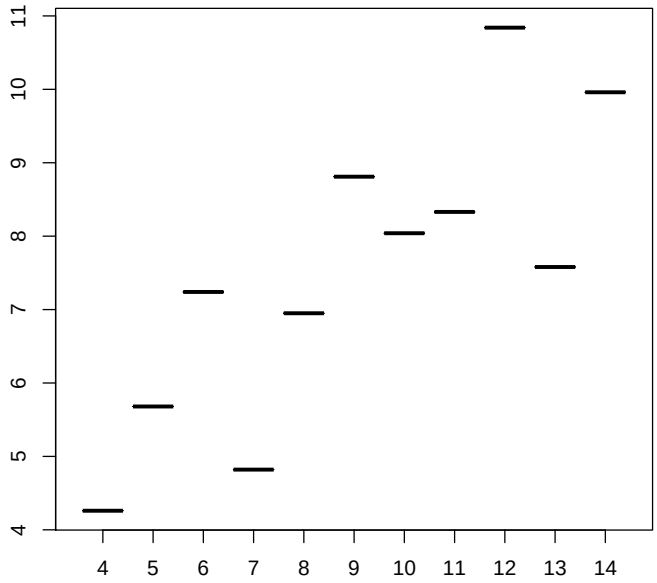
```
plot(as.factor(dados$x1))
```



Boxplot

Se dois vectores, um do tipo factor e outro númerico, são fornecidos a função plot gera boxplot.

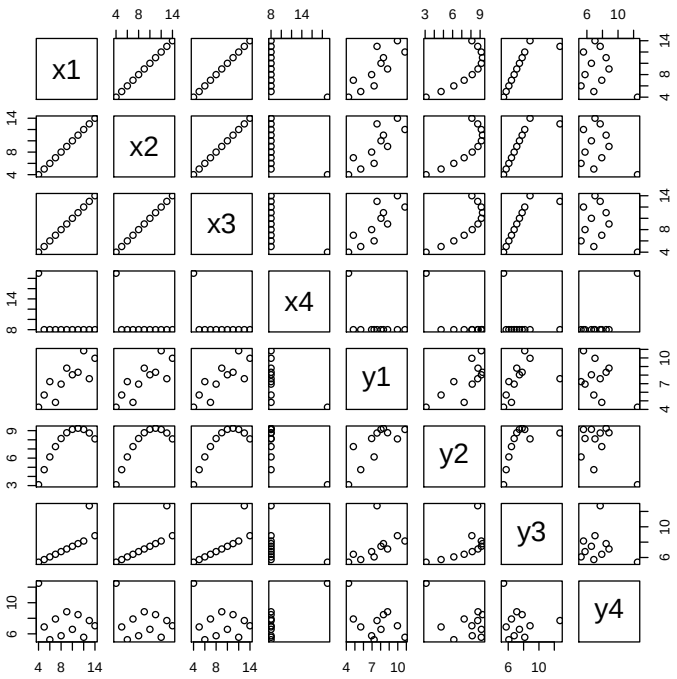
```
plot(as.factor(dados$x1), dados$y1)
```



Painéis gráficos

Se um dataframe é fornecido a função plot gera pares de gráficos de pontos para todas as variáveis.

```
plot(dados)
```



2.2 Função par()

Os parâmetros gráficos podem ser acessados pela ajuda ?par. O R possui vários dispositivos que são usados para gerar os gráficos. Normalmente usa-se um dispositivo para visualizar os gráficos e outro para exportar os gráficos para um arquivo. Informações sobre os dispositivos podem ser obtidas na ajuda ?Devices. Cada dispositivo possui uma lista de parâmetros gráficos

```
ls(par())  # Parâmetros padrão do dispositivo ativo
```

```
##  [1] "adj"       "ann"       "ask"       "bg"        "bty"       "cex"       "cex.axis"
##  [8] "cex.lab"   "cex.main"  "cex.sub"   "cin"       "col"       "col.axis"  "col.lab"
## [15] "col.main"  "col.sub"   "cra"       "crt"       "csi"       "cxy"       "din"
## [22] "err"       "family"    "fg"        "fig"       "fin"       "font"      "font.axis"
## [29] "font.lab"  "font.main" "font.sub"  "lab"       "las"       "lend"      "lheight"
## [36] "ljoin"     "lmitre"    "lty"       "lwd"       "mai"       "mar"       "mex"
## [43] "mfcol"     "mfg"       "mfrow"     "mgp"       "mkh"       "new"       "oma"
## [50] "omd"       "omi"       "page"      "pch"       "pin"       "plt"       "ps"
## [57] "pty"       "smo"       "srt"       "tck"       "tcl"       "usr"       "xaxp"
## [64] "xaxs"      "xaxt"      "xlog"      "xpd"       "yaxp"      "yaxs"      "yaxt"
## [71] "ylbias"    "ylog"
```

Cada parâmetro define uma opção de configuração do gráfico. Por exemplo, o parâmetro pch define o tipo de símbolo que é usado em um gráfico de pontos. Os parâmetros definem uma infinidade de opções como cor de texto e tamanho e margem. Pode-se alterar os parâmetros gráficos de duas formas.

A primeira maneira de alterar um parâmetro gráficos é expecificando o nome do parâmetro e o novo valor que receberá diretamente da função plot. Desta maneira o gráfico que será construido mostrará a nova opção, conforme especificado. Uma segunda maneira de alterar é especificar na função par antes de começar a construir o gráfico. Isso afetará de maneira permanete os todos os gráficos feitos pelo dispositivo, até o dispositivo ser reiniciado.

Pode-se fazer uma cópia de todos os parâmetros gráficos do dispositivo em um objeto. Depois que alterado os parâmetros pela função par pode-se restaurar a qualquer momento.

```
resetPar <- par()  # Fazer cópia dos parâmetros.
par(resetPar)  # Restaurar parâmetros originais.
```

Alguns parâmetros do dispositivo não podem ser alterados. Uma mensagem alertará sobre este fato, mas ela pode ser ignorada.

2.3 Alterando eixos

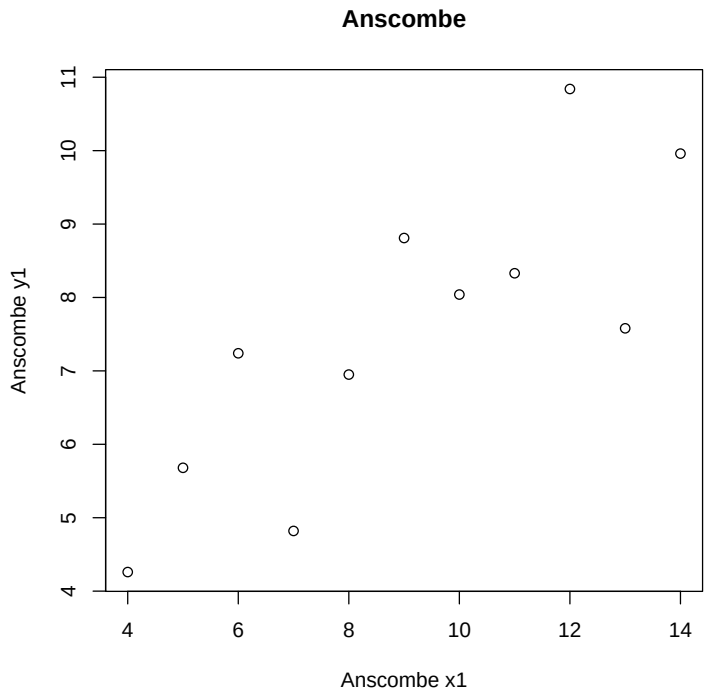
As opções mais usadas na elaboração de um gráfico é a alteração de aspectos relacionados aos eixos de um gráfico. Em quase todos os gráficos altera-se os nomes dos eixos, título do gráfico e os limites dos eixos.

Argumentos:

xlab e **ylab** - Alterar nomes dos eixos x e y respectivamente. Podem ser usados junto com font.lab para alterar o tipo de fonte, col.lab para alterar cor e cex.lab para alterar o tamanho da fonte.

main - Alterar nomes do título do gráfico. Pode ser usado junto com font.main, col.main e cex.main, para alterar fonte, cor e tamanho do título respectivamente.

```
plot(dados$x1, dados$y1,
    xlab = "Anscombe x1", ylab = "Anscombe y1",
    main = "Anscombe")
```



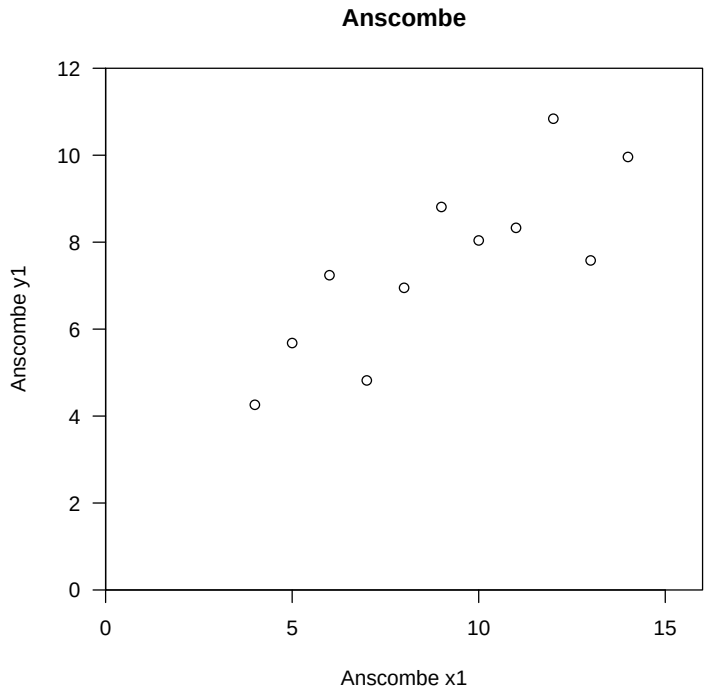
las - Alterar orientação do nomes dos eixos. Valores aceitos 0 (paralelo ao eixo), 1 (horizontal), 2 (perpendicular) e 3 (vertical).

xlim e **ylim** - Alterar limites dos eixos x e y respectivamente. Argumento do tipo vector, com dois valores, sendo o primeiro o mínimo e o segundo o valor máximo do eixo.

xaxs e **yaxs** - Alterar o estilo de cálculo dos limites dos eixos x e y respecivamente. Valores aceitos "r"(o limite de cada eixo estende por quatro porcento em cada lado do eixo) e "i"(o limite fica exatamente no estabelecidos pelos argumentos xlim e ylim ou pelos dados).

cex.axis - Altera o tamanho da fonte para os valores dos eixos. Argumento numérico com padrão = 1. Pode ser usado junto com font.axis para alterar o tipo de fonte, col.axis para alterar a cor.

```
plot(dados$x1, dados$y1,
    xlab = "Anscombe x1", ylab = "Anscombe y1",
    main = "Anscombe",
    las = 1,
    xlim = c(0, 16), ylim = c(0, 12),
    xaxs = "i", yaxs = "i",
    cex.axis = 1)
```



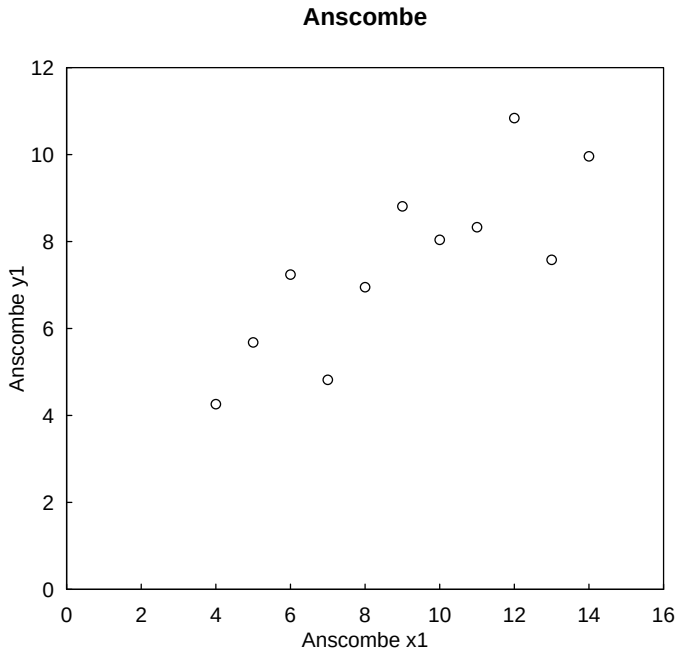
xaxp e **yaxp** - Alteram posiçao e quantidade de separadores dos eixos x e y respecivamente. Argumento do tipo vector do tipo c(x1, x2, n), onde x1 e x2, mínimo e máximo, que representa o alcance do separadores e n a quantidade espaços internos.

tck - Altera o comprimento dos separados dos eixos. Argumento númerico, valores expressos em fração do tamanho do eixo. Se positivo marcadores fica na parte interna do gráfico, se negativos ficam na parte externa e se zero separadores não são mostrados.

mgp - Altera a linha da margem onde são mostrados os textos do eixos, valores dos eixos e próprio eixo. Argumento do tipo vector com padrão c(3, 1, 0). O primeiro valor altera texto do eixo, o segundo altera os valores dos eixos e o terceiro linha no próprio eixo. Valores próximos de 0 aproximam os textos e valores ao eixo. Pode ser usado valores negativos.

```
plot(dados$x1, dados$y1,
    xlab = "Anscombe x1", ylab = "Anscombe y1",
    main = "Anscombe",
    las = 1,
    xlim = c(0, 16), ylim = c(0, 12),
    xaxs = "i", yaxs = "i",
    cex.axis = 1,
    xaxp = c(0, 16,8), yaxp = c(0, 12, 6),
    tck = 0.01,
```

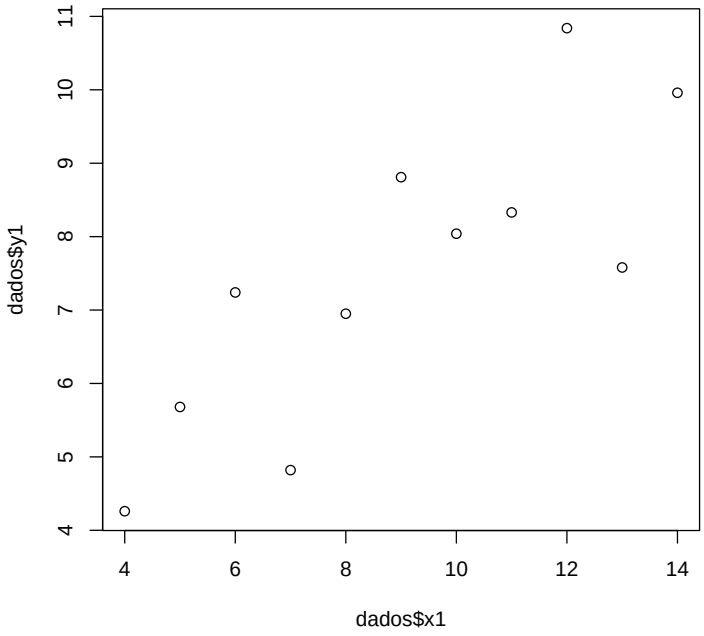
```
    mgp = c(1.5, 0.5, 0))
```



2.4 Descontruindo o gráfico

Assim como elementos podem ser adicionados ao gráficos, eles também podem ser ocultados no gráfico.

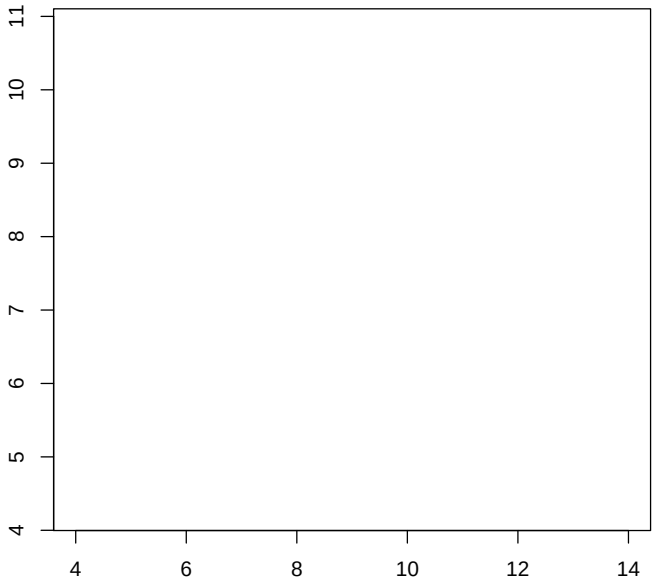
```
plot(dados$x1, dados$y1)
```



Argumentos:

type - Altera o tipo de gráfico que será construido. Vários valores são aceito, os principais são "p"(gráfico de pontos), "l"(gráfico de linhas), "b"(gráficos com pontos e linhas) e "n"(para não mostrar nada referende aos dados).

```
plot(dados$x1, dados$y1,
    type = "n",
    xlab = "", ylab = "")
```



xaxt e **yaxt** - Expecifica se o eixo x e y, respectivamente, deve ser mostrado. Quando o valor for "n"o eixo não é mostrado.

```
plot(dados$x1, dados$y1,
    type = "n",
    xlab = "", ylab = "",
    xaxt = "n", yaxt = "n")
```

bty - Altera o tipo de caixa (box) usado envolta do gráfico. Valores aceito "o", "l", "7", "c", "u"e "]"que resultam em contorno igual a forma do símbolo ou "n"para não mostrar contorno.

```
plot(dados$x1, dados$y1,
    type = "n",
    xlab = "", ylab = "",
    xaxt = "n", yaxt = "n",
    bty = "n")
```

A figura acima ficou totalmente branca, sem elementos visíveis. As margens e limites dos eixos são os mesmos que o gráfico original, embora todos os elementos foram ocultos.

2.5 Pontos

A função auxiliar points adiciona pontos ou linhas ao gráfico.

Argumentos:

x e **y** - Valor único com vector com as coordenadas dos pontos para o eixo x e y respectivamente.

pch - Altera o tipo de símbolo dos pontos. Valores aceitos entre 0 e 25. Se for um único valor todos os símbolos serão idênticos, se for um vector cada ponto receberá um símbolo conforme especificado.

cex - Altera o tamanho do símbolos. Argumento com valor numérico com padrão = 1. Pode ser um vector com um valor para cada ponto.

```
plot(dados$x1, dados$y1,
    type = "n",
    xlab = "", ylab = "",
    xaxt = "n", yaxt = "n",
    bty = "n")

points(x = dados$x1, y = dados$y1,
    pch = 23,
    cex = 1.3)
```

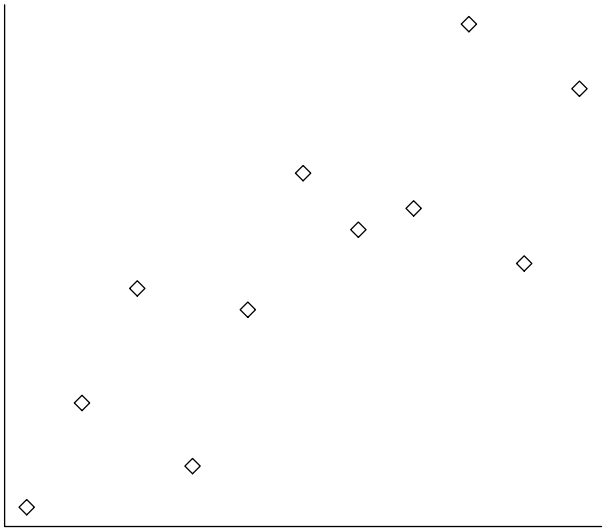
2.6 Box

A função auxiliar box pode adicionar linhas de contorno em um gráfico.

Argumento:

bty" - Altera o tipo de caixa (box) usado envolta do gráfico. Valores aceito "o", "l", "7", "c", "u"e "]"que resultam em contorno igual a forma do símbolo ou "n"que não mostra contorno.

```
plot(dados$x1, dados$y1,
    type = "n",
    xlab = "", ylab = "",
    xaxt = "n", yaxt = "n",
    bty = "n")

points(x = dados$x1, y = dados$y1,
    pch = 23,
    cex = 1.3)

box(bty = "l")
```



2.7 Eixos

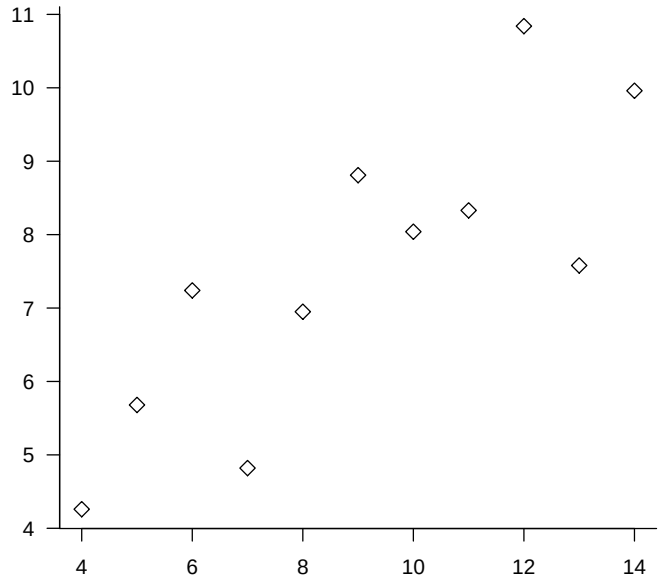
Função auxiliar axis pode adicionar eixos ao gráfico.

Argumentos:

side - Especificar a posição do eixo. Argumento numérico com os valores de 1 (eixo inferior), 2 (eixo à esquerda), 3 (eixo superior) e 4 (eixo à direita).

las - Alterar orientação do rótulos do eixo. Valores aceitos 0 (paralelo ao eixo), 1 (horizontal), 2 (perpendicular) e 3 (vertical).

```
plot(dados$x1, dados$y1,
    type = "n",
    xlab = "", ylab = "",
    xaxt = "n", yaxt = "n",
    bty = "n")

points(x = dados$x1, y = dados$y1,
    pch = 23,
    cex = 1.3)

box(bty = "l")

axis(side = 1)

axis(side = 2,
    las = 1)
```



2.8 Linhas

Função auxiliar lines adiciona linhas a um gráfico. Pode ser usada para adicionar linhas de modelos lineares e não lineares.

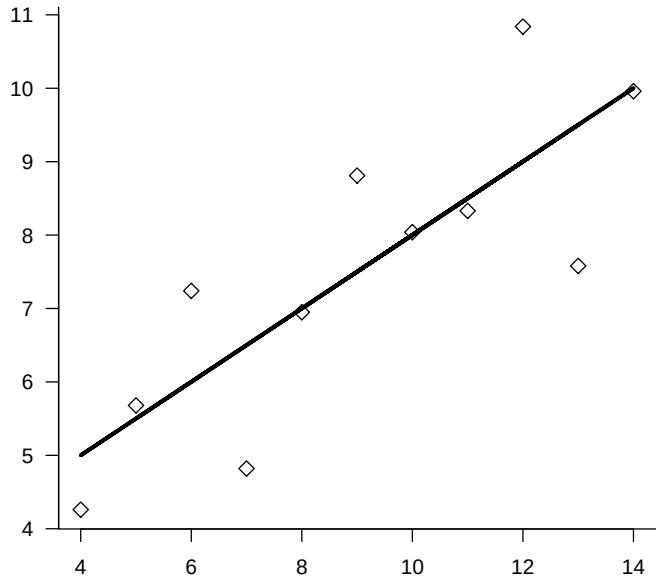
Argumentos:

lty - Altera o tipo de linha. Valores aceitos 0 (sem linha), 1 (linha sólida), 2 (linha tracejada), 3 (linha pontilhada), 4 (pontilhada e tracejada), 5 (linha com traço longo) e 6 (linha com traço duplo). Alternativamente pode ser usado os valores "blank", "solid", "dashed", "dotted", "dotdash", "longdash"e "twodash", que substitui dos números de 1 a 6 respectivamente.

lwd - Altera a espessura da linha. Argumento númerico com padrão = 1.

```
mod <- lm(dados$y1 ~ dados$x1)
mod # Modelo linear

##
## Call:
## lm(formula = dados$y1 ~ dados$x1)
##
## Coefficients:
## (Intercept)     dados$x1
##         3.0          0.5

ypredito <- predict(mod)
ypredito # Valores preditos pelos modelo linear

##      1      2      3      4      5      6      7      8      9     10     11
##  8.001  7.001  9.501  7.501  8.501 10.001  6.001  5.000  9.001  6.501  5.501


plot(dados$x1, dados$y1,
```

```
    type = "n",
    xlab = "", ylab = "",
    xaxt = "n", yaxt = "n",
    bty = "n")

points(x = dados$x1, y = dados$y1,
    pch = 23,
    cex = 1.3)

box(bty = "l")

axis(side = 1)

axis(side = 2,
    las = 1)

lines(x = dados$x1, y = ypredito,
    lty = 1,
    lwd = 3)
```



2.9 Legendas

A função legend adiciona legenda ao gráfico.

Argumentos:

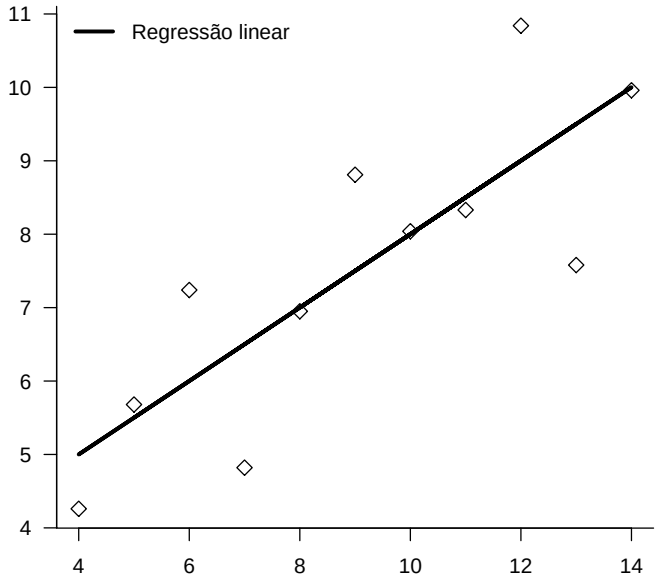
x e **y** - Coordenadas para a posição da legenda. Alternativamente pode ser substituido pelos valores "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right"e "center"para usar posições predefinidas.

legend - Texto que mostrado na legenda. Pode ser uma única expressão ou uma vector, sendo que cada um representa um item na legenda.

lty, **lwd**, **pch**, **angle** e **density** - Especificar o tipo de símbolo usado na legenda. Os argumentos são os mesmo usado na construção de linhas (lty e lwd), pontos (pch) e barras (angle e density).

bty - Altera o tipo de caixa (box) usado envolta da legenda. Valores aceito "o"(todo contorno) e "n"(sem contorno).

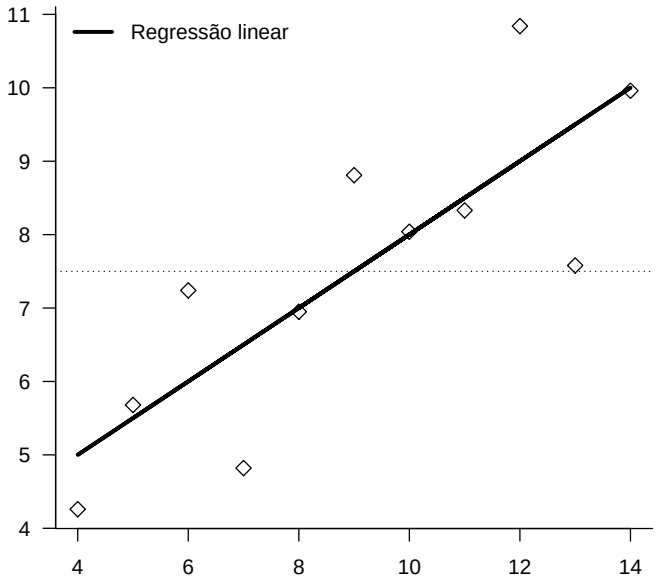
```
plot(dados$x1, dados$y1,
    type = "n",
    xlab = "", ylab = "",
    xaxt = "n", yaxt = "n",
    bty = "n")

points(x = dados$x1, y = dados$y1,
    pch = 23,
    cex = 1.3)

box(bty = "l")

axis(side = 1)

axis(side = 2,
    las = 1)

lines(x = dados$x1, y = ypredito,
    lty = 1,

legend(x = "topleft",
    legend = "Regressão linear",
    lty = 1,
    lwd = 3,
    bty = "n")
```



2.10 Linhas retas

Função auxiliar abline é usada para adicionar linhas simples aos gráficos.
Argumentos:
h e **v** - Especificar valor para linha horizontal e vertical respectivamente.

```
plot(dados$x1, dados$y1,
    type = "n",
    xlab = "", ylab = "",
    xaxt = "n", yaxt = "n",
    bty = "n")

points(x = dados$x1, y = dados$y1,
    pch = 23,
    cex = 1.3)

box(bty = "l")

axis(side = 1)

axis(side = 2,
    las = 1)

lines(x = dados$x1, y = ypredito,
    lty = 1,

legend(x = "topleft",
    legend = "Regressão linear",
    lty = 1,
    lwd = 3,
    bty = "n")

abline(h = 7.5,
    lty = 3)
```



2.11 Setas

Função auxiliar arrows permite adicionar setas ao gráficos.

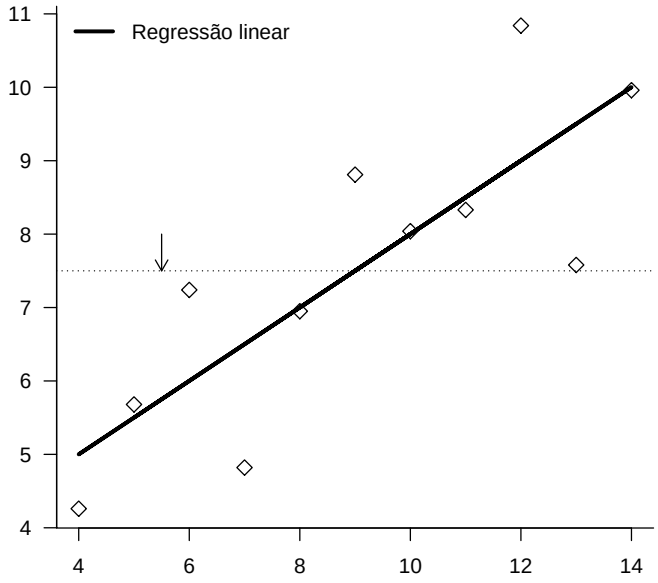
Argumentos:

x0, **y0**, **x1** e **y1** - Coordenadas para o início e fim da seta. Sendo que x0 e y0 são para a origem da seta e x1 e y1 para o destino da seta. Pode ser um único valor ou um vector para adicionar várias setas.

length - Alterar o comprimento da ponta da seta. Argumento númerico expresso em polegadas com padrão = 0.25.

```
plot(dados$x1, dados$y1,
    type = "n",
    xlab = "", ylab = "",
    xaxt = "n", yaxt = "n",
    bty = "n")

points(x = dados$x1, y = dados$y1,
    pch = 23,
    cex = 1.3)

box(bty = "l")

axis(side = 1)

axis(side = 2,
    las = 1)

lines(x = dados$x1, y = ypredito,
    lty = 1,
```

```
legend(x = "topleft",
    legend = "Regressão linear",
    lty = 1,
    lwd = 3,
    bty = "n")

abline(h = 7.5,
    lty = 3)

arrows(x0 = 5.5, y0 = 8,
    x1 = 5.5, y1 = 7.5,
    length = 0.1)
```



2.12 Textos

Função auxiliar text permite adicionar texto aos gráfico.

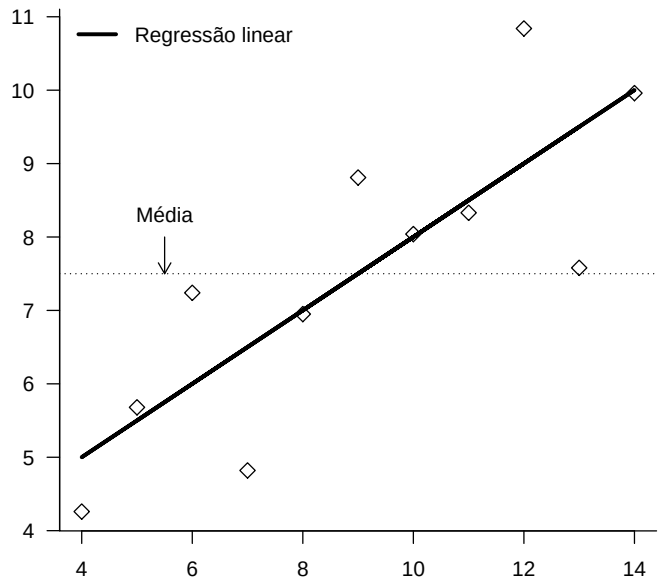
Argumentos:

x e **y** - Coordendas para o texto. Pode ser um vector com uma série de valores

labels - Texto ou expressão para adicionar a coordenada. Pode ser um vector com o texto de será escrito para cada coordenada fornecida.

```
plot(dados$x1, dados$y1,
    type = "n",
    xlab = "", ylab = "",
    xaxt = "n", yaxt = "n",
    bty = "n")

points(x = dados$x1, y = dados$y1,
```

```
    pch = 23,
    cex = 1.3)

box(bty = "l")

axis(side = 1)

axis(side = 2,
    las = 1)

lines(x = dados$x1, y = ypredito,
    lty = 1,

legend(x = "topleft",
    legend = "Regressão linear",
    lty = 1,
    lwd = 3,
    bty = "n")

abline(h = 7.5,
    lty = 3)

arrows(x0 = 5.5, y0 = 8,
    x1 = 5.5, y1 = 7.5,
    length = 0.1)

text(x = 5.5, y = 8.3,
    labels = "Média")
```



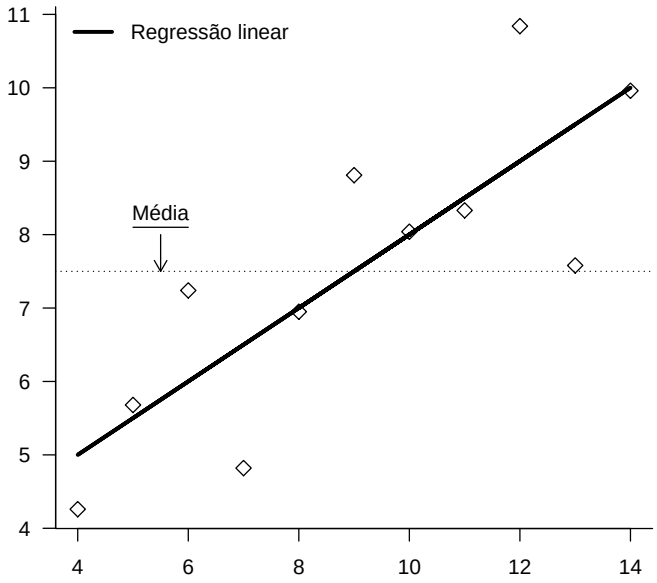
2.13 Segmentos

Função auxiliar segments permite adicionar pequenos segmentos ao gráfico.

Argumentos:

x0, **y0**, **x1** e **y1** - Coordenadas para o segmento o início e o fim do segmento. Pode ser um único valor ou um vector para adicionar vários segmentos.

```
plot(dados$x1, dados$y1,
    type = "n",
    xlab = "", ylab = "",
    xaxt = "n", yaxt = "n",
    bty = "n")

points(x = dados$x1, y = dados$y1,
    pch = 23,
    cex = 1.3)

box(bty = "l")

axis(side = 1)

axis(side = 2,
    las = 1)

lines(x = dados$x1, y = ypredito,
    lty = 1,

legend(x = "topleft",
    legend = "Regressão linear",
    lty = 1,
    lwd = 3,
    bty = "n")

abline(h = 7.5,
    lty = 3)

arrows(x0 = 5.5, y0 = 8,
    x1 = 5.5, y1 = 7.5,
    length = 0.1)

text(x = 5.5, y = 8.3,
    labels = "Média")

segments(x0 = 5, y0 = 8.1,
    x1 = 6, y1 = 8.1)
```



2.14 Grid

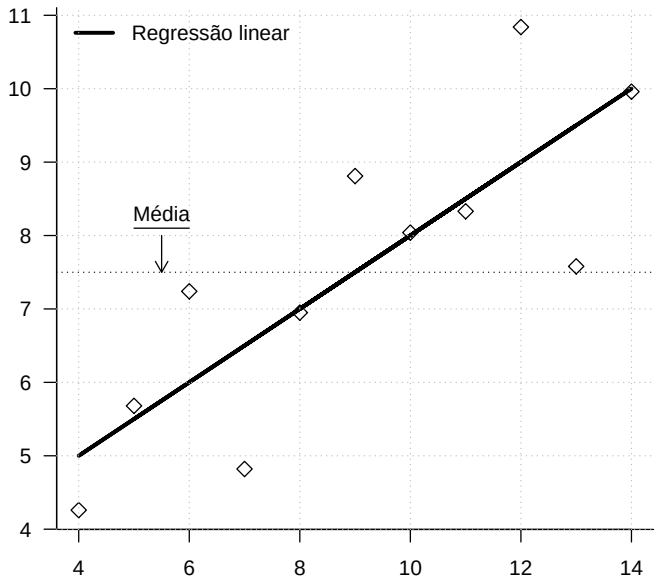
Função auxiliar grid permite adicionar um grid ao gráfico.

Argumentos:

nx e **ny** - Especificar o número de células do grid para as direções de x e y respectivamente. Se for NULL as linhas são mostrada nos marcadores de cada eixo.

```
plot(dados$x1, dados$y1,
    type = "n",
    xlab = "", ylab = "",
    xaxt = "n", yaxt = "n",
    bty = "n")

points(x = dados$x1, y = dados$y1,
    pch = 23,
    cex = 1.3)

box(bty = "l")

axis(side = 1)

axis(side = 2,
    las = 1)

lines(x = dados$x1, y = ypredito,
    lty = 1,

legend(x = "topleft",
    legend = "Regressão linear",
```

```
    lty = 1,
    lwd = 3,
    bty = "n")

abline(h = 7.5,
    lty = 3)

arrows(x0 = 5.5, y0 = 8,
    x1 = 5.5, y1 = 7.5,
    length = 0.1)

text(x = 5.5, y = 8.3,
    labels = "Média")

segments(x0 = 5, y0 = 8.1,
    x1 = 6, y1 = 8.1)

grid(nx = NULL, ny = NULL,
    lty = 3,
    lwd = 1,
    col = "gray")
```



2.15 Títulos

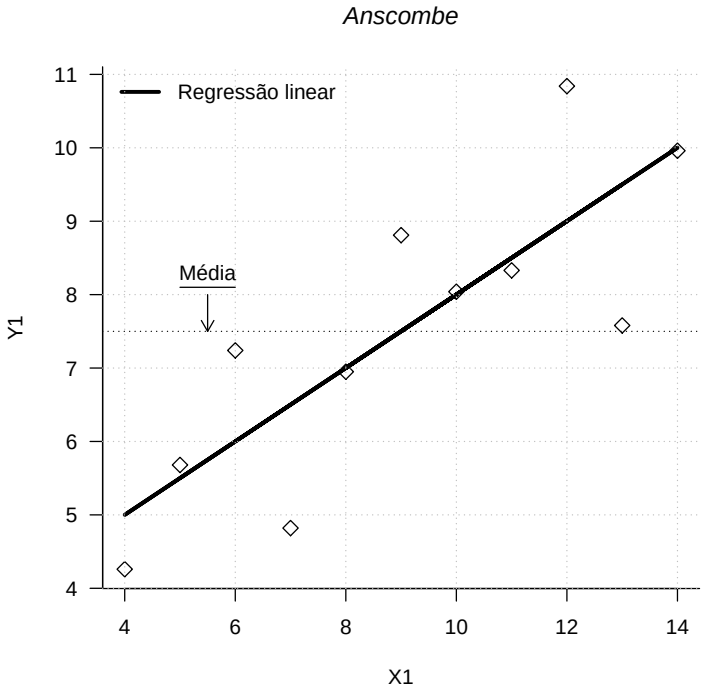
A função auxiliar title permite adicionar títulos ao gráfico e aos eixos.

Argumentos:

main, **xlab** e **ylab** - Texto para o título do gráfico e para os rótulos dos eixos x e y, respectivamente.

font - Especificar tipo de fonte. O tipo de fonte depende do dispositivo usado. De uma maneira básica 1 (fonte normal), 2 (fonte em negrito), 3 (fonte em itálico) e 4 (fonte em negrito e itálico).

```r
plot(dados$x1, dados$y1,
    type = "n",
    xlab = "", ylab = "",
    xaxt = "n", yaxt = "n",
    bty = "n")

points(x = dados$x1, y = dados$y1,
    pch = 23,
    cex = 1.3)

box(bty = "l")

axis(side = 1)

axis(side = 2,
    las = 1)

lines(x = dados$x1, y = ypredito,
    lty = 1,

legend(x = "topleft",
    legend = "Regressão linear",
    lty = 1,
    lwd = 3,
    bty = "n")

abline(h = 7.5,
    lty = 3)

arrows(x0 = 5.5, y0 = 8,
    x1 = 5.5, y1 = 7.5,
    length = 0.1)

text(x = 5.5, y = 8.3,
    labels = "Média")

segments(x0 = 5, y0 = 8.1,
    x1 = 6, y1 = 8.1)

grid(nx = NULL, ny = NULL,
    lty = 3,
    lwd = 1,
    col = "gray")

title(main = "Anscombe",
    xlab = "X1", ylab = "Y1",
    font.main = 3)
```



2.16 Função locator()

Muitas funções usadas para adicionar itens aos gráficos precisam da localização exata no gráfico. Uma opção para obter a localização é usar a função locator(). Executamdo a função locator() ela permite clicar na área do gráfico e obter as coordenadas do ponto. A função fecha automaticamente se for clicado no botão direito do mouse ou expecificando o argumento n com a quantidade de pontos que serão obtido.

```
locator(n = 1, labels = "Cliquei aqui")
```

2.17 Expressões

Elementos com formatação especiais e notaçoes matemáticas, letras gregas podem ser mostrados na forma de texto, títulos e eixos usando a função expression (ver ?expression). Mais informações podem ser encontradas na função plotmath (ver ?plotmath) e em demo(plotmath).

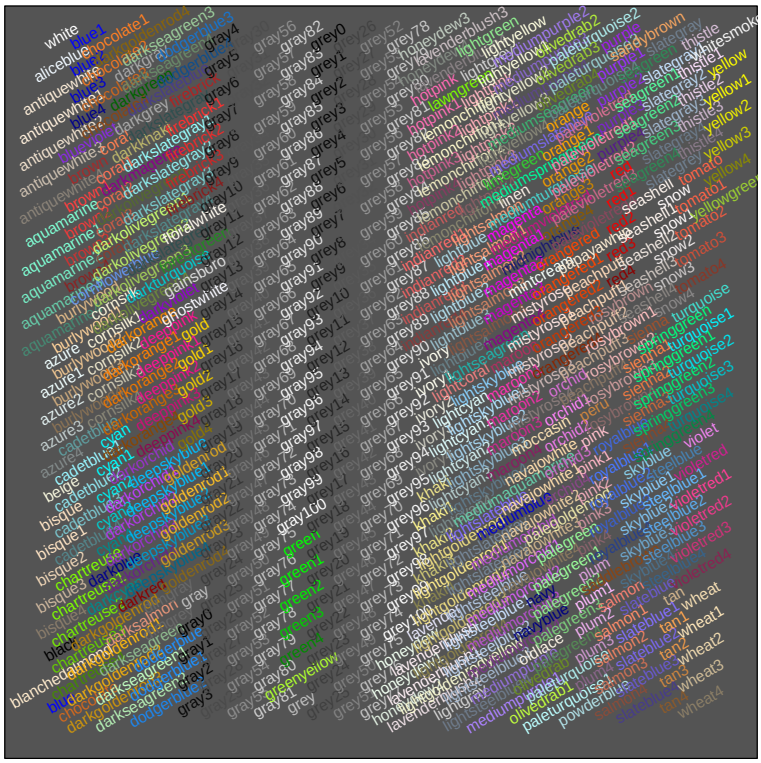
Abaixo segue exemplo de expressões que podem ser usadas. As figuras foram geradas a partir do exemplos de demo(plotmath).

x + y	$x + y$	x %subset% y	$x \subset y$
x − y	$x - y$	x %subseteq% y	$x \subseteq y$
x * y	xy	x %supset% y	$x \supset y$
x/y	x/y	x %supseteq% y	$x \supseteq y$
x %+−% y	$x \pm y$	x %notsubset% y	$x \not\subset y$
x%/%y	$x \div y$	x %in% y	$x \in y$
x %*% y	$x \times y$	x %notin% y	$x \notin y$
x %.% y	$x \cdot y$	hat(x)	$\hat{x}$
x[i]	x_i	tilde(x)	$\tilde{x}$
x^2	x^2	ring(x)	$\mathring{x}$
paste(x, y, z)	xyz	bar(xy)	$\overline{xy}$
sqrt(x)	$\sqrt{x}$	widehat(xy)	$\widehat{xy}$
sqrt(x, y)	$\sqrt[y]{x}$	widetilde(xy)	$\widetilde{xy}$
list(x, y, z)	x, y, z	x %<−>% y	$x \leftrightarrow y$
x == y	$x = y$	x %−>% y	$x \rightarrow y$
x != y	$x \neq y$	x %<−% y	$x \leftarrow y$
x < y	$x < y$	x %up% y	$x \uparrow y$
x <= y	$x \leq y$	x %down% y	$x \downarrow y$
x %~~% y	$x \approx y$	x %<=>% y	$x \Leftrightarrow y$
x %=~% y	$x \cong y$	x %=>% y	$x \Rightarrow y$
x %==% y	$x \equiv y$	x %<=% y	$x \Leftarrow y$
x %prop% y	$x \propto y$	x %dblup% y	$x \Uparrow y$
x %~% y	$x \sim y$	x %dbldown% y	$x \Downarrow y$
plain(x)	x	Alpha − Omega	$A - \Omega$
italic(x)	x	alpha − omega	$\alpha - \omega$
bold(x)	$\mathbf{x}$	phi1 + sigma1	$\varphi + \varsigma$
bolditalic(x)	$\boldsymbol{x}$	Upsilon1	Υ
underline(x)	$\underline{x}$	infinity	∞
list(x[1], ..., x[n])	$x_1, \ldots, x_n$	32 * degree	32°
x[1] + ... + x[n]	$x_1 + \cdots + x_n$	60 * minute	$60'$
list(x[1], cdots, x[n])	$x_1, \cdots, x_n$	30 * second	$30''$
x[1] + ldots + x[n]	$x_1 + \ldots + x_n$	x ~ ~y	$x \ \ y$

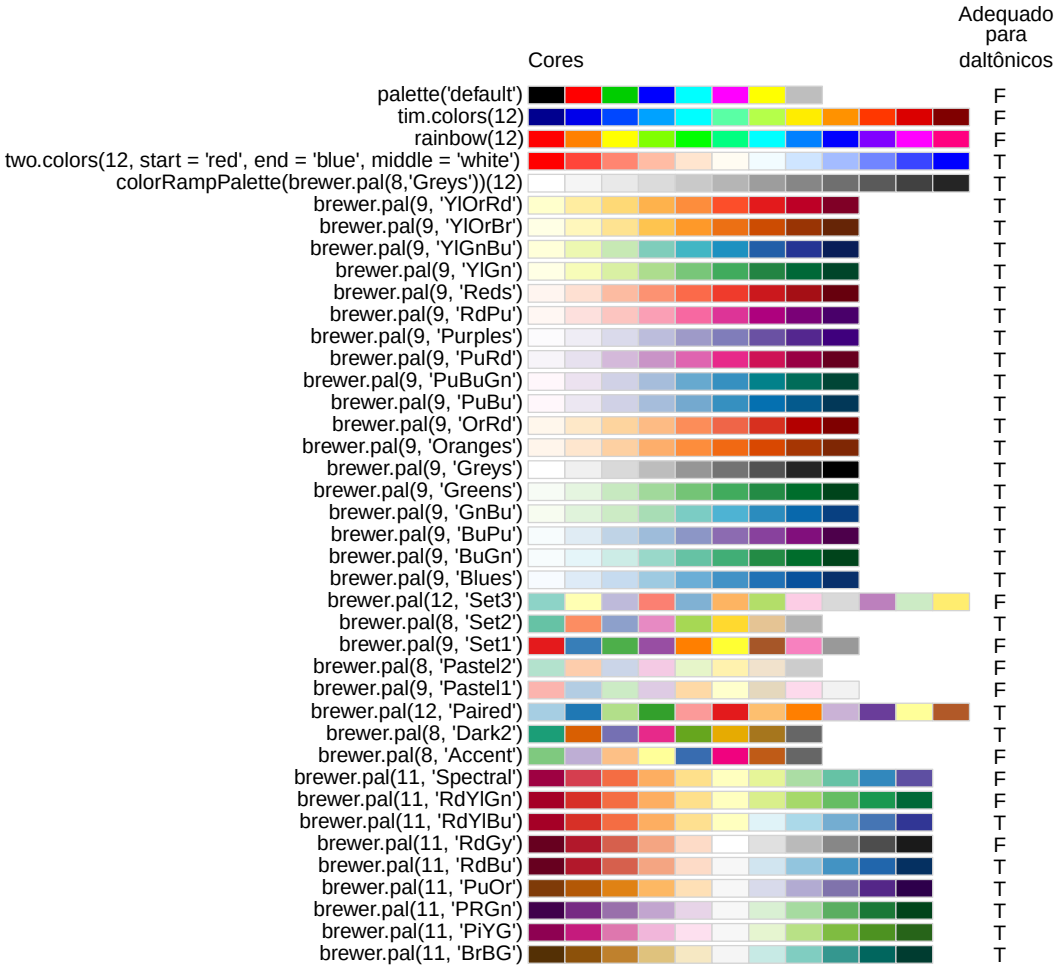
frac(x, y)	$\frac{x}{y}$	min(g(x), x >= 0)	$\min_{x \geq 0} g(x)$
over(x, y)	$\frac{x}{y}$	inf(S)	$\inf S$
atop(x, y)	$\begin{matrix} x \\ y \end{matrix}$	sup(S)	$\sup S$
sum(x[i], i = 1, n)	$\sum_{1}^{n} x_i$	(x + y) * z	$(x + y) z$
prod(plain(P)(X == x), x)	$\prod_{x} P(X = x)$	x^y + z	$x^{y} + z$
integral(f(x) * dx, a, b)	$\int_{a}^{b} f(x) dx$	x^(y + z)	$x^{(y+z)}$
union(A[i], i == 1, n)	$\bigcup_{i=1}^{n} A_i$	x^{y + z}	x^{y+z}
intersect(A[i], i == 1, n)	$\bigcap_{i=1}^{n} A_i$	bgroup("(", atop(x, y), ")")	$\binom{x}{y}$
lim(f(x), x %−>% 0)	$\lim_{x \to 0} f(x)$	group("|", x, "|")	$\lvert x \rvert$

2.18 Cores

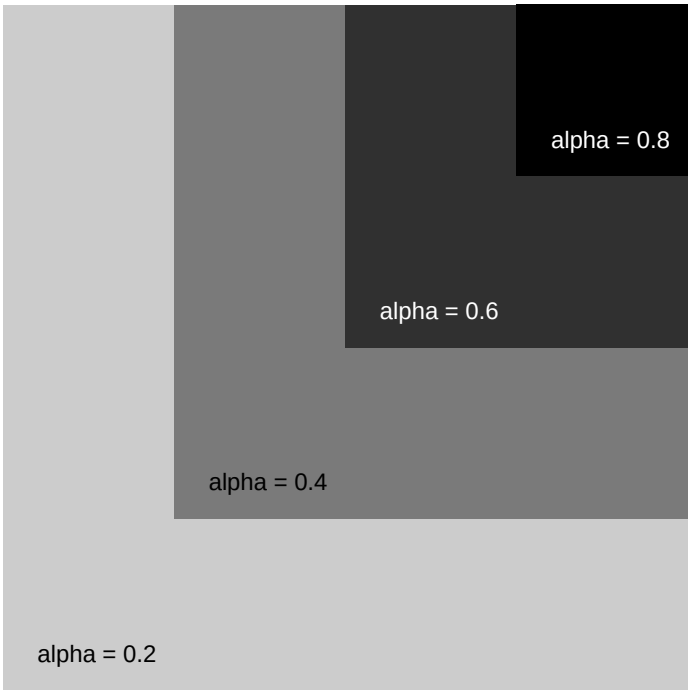
As cores de cada um dos itens podem ser alteradas pelo argumento cor. As cores básicas para a maioria dos gráficos são 'black', 'white' e 'grey'. Mais informações sobre cores podem ser buscadas na ajuda de ?colors e demo(colors). Ao fazer gráficos coloridos certifique que as cores que estiver usando são apropriadas para daltônicos. A figura a seguir foi gerada pelo demo da função demo(colors).



Existem algumas paletas de cores prontas no R. A paleta básica pode ser encontrada com a função palette(). Essa paleta é composta por apenas 8 cores, mas essa pode ser alterada permitindo adicional, remover ou personalizar as a paleta de cores. Os pacotes fields e RColorBrewer oferecem algumas funções interessantes para criar paletas. Essas paletas podem ser expandidas usando a função colorRampPalette. A figura a seguir mostra algumas opções de comandos que podem ser utilizados para gerar paletas de cores.



Você pode ajustar a opacidade das cores usando a função adjustcolor alterando o argumento alpha. Na figura a seguir foi usado apenas a cor 'black' e alteradas os parâmetros alpha. Valores de alpha próximos de 0 deixar as cores mais opacas (transparências) enquanto que valores próximos a 1 deixas as cores sólidas. Gráficos feitos com essas opções permitem sobrepor informações em uma mesma figura.



2.19 Fontes

Cada dispositivo gráfico do R possui um conjunto de fontes disponível. A lista de fontes pode ser acessada pelas funções: quartzFonts() para OS X, X11Fonts() para Linux, windowsFonts() para Windows, pdfFonts() para pdf e postscriptFonts() para postscript. As fontes podem ser alteradas pelo argumentos family e font.

2.20 Formas geómetricas

Para adicionar formas geómetricas ao gráfico pode-se usar as funções do pacote shape. As funções mais simples são: filledrectangle (para retangulos e quadrados), filledellipse (para elipses e círculos) e filledmultigonal (para outros objetos com 3 ou mais lados).

Argumentos:

mid - Especificar centro da forma geométrica. Argumento do tipo c(x, y), com as coordenadas para o eixo x e y respectivamente.

wx e **wy** - Especificar largura horizontal e vertical do retângulo, respectivamente.

rx e **ry** (**rx1** e **ry1**) - Especificar raio horizontal e vertical, respectivamente.

nr - Especificar número de lados do polígono.

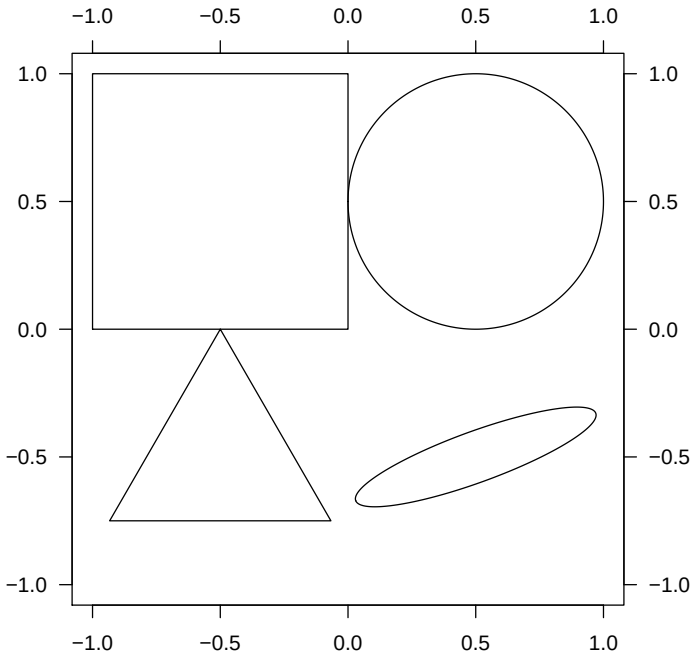
angle - Especificar ángulo de rotação em graus.

col - Especificar cor para o preenchimento.

density - Especificar densidade de linhas de sombreamento, em linhas por polegada.

Quando col = "black"e density = 0 a forma geométrica fica sem preenchimento. Para definir o tipo da borda é preciso definir antes de começar o gráfico, na função par.

```r
require(shape)

par(mar = c(4, 4, 4, 4), las = 1) # Alterar tamanho das margens.

plot(1, 1,
    xlim = c(-1, 1), ylim = c(-1, 1),
    type = "n",
    xlab = "", ylab = "")

axis(3)
axis(4)

filledrectangle(mid = c(-0.5, 0.5),
    wx = 1, wy = 1,
    col = "black",
    density = 0)

filledellipse(rx1 = 0.5, ry1 = 0.5,
    mid = c(0.5, 0.5),
    col = "black",
    density = 0)

filledmultigonal(mid = c(-0.5, -0.5),
    rx = 0.5,
    nr = 3,
    col = "black",
    angle = 90,
    density = 0)

filledellipse(rx1 = 0.5,
    ry1 = 0.1,
    mid = c(0.5, -0.5),
    col = "black",
    angle = 20,
    density = 0)

par(resetPar) # Restaurar parâmetros originais.
```



3 Gráficos

Nesta seção o objetivo é mostrar alguns tipos de gráficos simples que podem ser contruídos a partir do R. Nem todos os elementos gráficos são aplicados em todos os gráfico. O objetivo é mostrar opções comuns encontrados em cada tipo de gráfico.

3.1 Gráfico de barras

Dados

Dados disponíveis no pacote ade4 com o nome de tortues. Os dados são 48 observações de tartarugas descritas por quatro variáveis (gênero, comprimento, largura e altura). Para cada gênero pode-se calcular média e desvio padrão das médidas.

```
require(ade4)
data(tortues)
str(tortues)

## 'data.frame': 48 obs. of  4 variables:
##  $ long: num  93 94 96 101 102 103 104 106 107 112 ...
##  $ larg: num  74 78 80 84 85 81 83 83 82 89 ...
##  $ haut: num  37 35 35 39 38 37 39 39 38 40 ...
##  $ sexe: Factor w/ 2 levels "M","F": 1 1 1 1 1 1 1 1 1 1 ...

dados <- rbind(tapply(tortues[, 1], tortues[, 4], mean),
    tapply(tortues[, 2], tortues[, 4],mean),
    tapply(tortues[, 1], tortues[, 4], mean))
rownames(dados) <- names(tortues)[1:3]
dados  # Média das medidas por gênero.
```

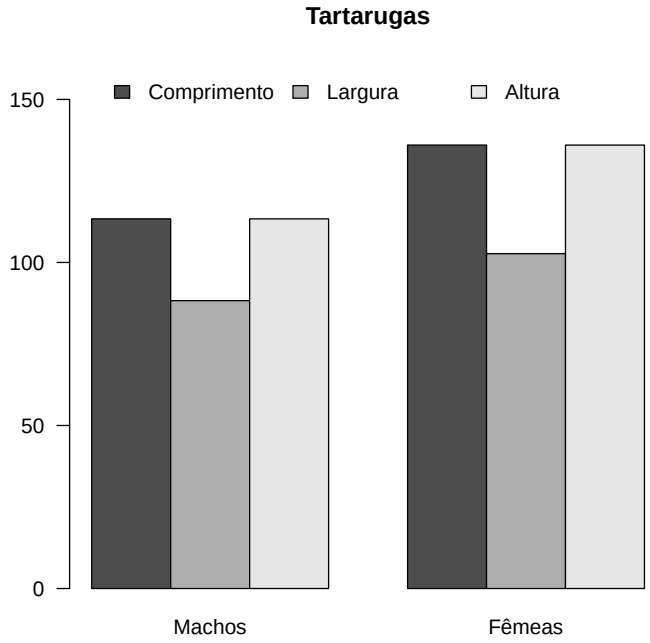
```
##           M     F
## long 113.38 136.0
## larg  88.29 102.7
## haut 113.38 136.0

dados_sd <- rbind(tapply(tortues[, 1], tortues[, 4], sd),
    tapply(tortues[, 2], tortues[, 4],sd),
    tapply(tortues[, 3], tortues[, 4], sd))
rownames(dados_sd) <- names(tortues)[1:3]
dados_sd  # Desvio padrão das medidas por gênero.

##           M      F
## long 11.780 21.246
## larg  7.074 13.113
## haut  3.352  8.465
```

Plot

A função barplot é expecifica para usar em gráficos de barras horizontais e verticais. Neste caso dos dados são uma pequena tabela com duas colunas e três linhas. Pode-se construir um gráfico mostrando seis barras com as medidas, agrupadas pelo gênero (pelas colunas). Neste gráfico as cores são definidas de maneira automática, mostradas em tons de cinza por padrão.

Argumentos:
width - Especificar largura das barras, com padrão de 1.
beside - Algumento lógico para especificar se colunas devem ser mostradas lado a lado.
legend - Especificar o texto de cada nível no gráfico.
args.legend - Especificar argumentos que serão ser usados na construção da legenda.
x - Parte de args.legend usado para especificar posição da legenda.
bty - Parte de args.legend usado para especificar tipo de borda da legenda.
ncol - Parte de args.legend usado para especificar número de colunas para os argumentos da legenda
names - Especificar nomes usados em cada grupo de barras.

```
barplot(dados,
    beside = TRUE,
    width = 0.7,
    ylim = c(0, 160),
    las = 1,
    legend = c("Comprimento","Largura", "Altura"),
    args.legend = list(x = "top", bty = "n", ncol = 3),
    names = c("Machos","Fêmeas"),
    main = "Tartarugas")
```



Para adicionar os linhas com o desvio padrão pode-se usar a função arrows (para setas). Uma alternativa é usar a função plotCI do pacote plotrix. A função permite adicionar barras de erros tanto na horizontal quanto na vertical. Um pequeno truque no caso do gráfico de barras e salvar os comandos do primeiro gráfico em um objeto qualquer (Neste caso objeto xx). O objeto terá a posição no eixo x de cada barra. A posição no eixo y são os próprios dados que foram usados para construir o gráfico. A altura de cada barra de erro foi calculada pelo desvio padrão em cada grupo.

Argumentos:

x e **y** - Coordenadas para o ínicio da cada linha de erro.

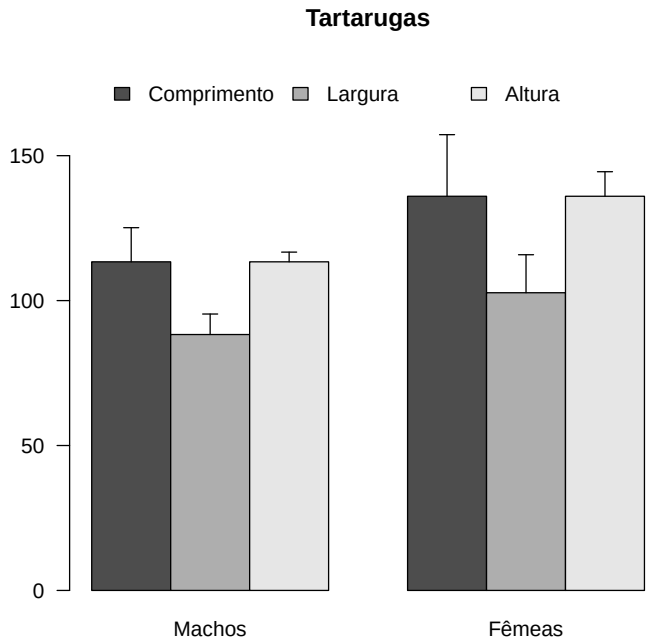
uiw e **liw** - Comprimento da linha de erro para a parte superior e inferior, respectivamente.

add - Argumento lógico para especificar se linhas de erro devem ser adicionadas ao gráfico existente ou não. Padrão é FALSE.

pch - Tipo de símbolo usado no centro da barra de erro. Se for NA não é mostrado nenhum símbolo.

```
require(plotrix)

xx <- barplot(dados,
    beside = TRUE,
    width = 0.7,
    ylim = c(0, 180),
    las = 1,
    legend = c("Comprimento","Largura", "Altura"),
    args.legend = list(x = "top", bty = "n", ncol = 3),
    names = c("Machos","Fêmeas"),
    main = "Tartarugas",
    xaxs = "r")
xx

##      [,1] [,2]
## [1,] 1.05 3.85
## [2,] 1.75 4.55
## [3,] 2.45 5.25
```

```
plotCI(x = xx,
    y = dados,
    uiw = dados_sd,
    liw = 0,
    add = TRUE,
    pch = NA)
```



3.2 Gráfico de barras empilhadas

Dados

Dados sobre a população brasileira na década de 2010. Os valores são em proporções para 4 categorias arbitrárias. Extraido do United Nations, Department of Economic and Social Affairs.

```
dados <- cbind(c(0.328, 0.461, 0.159, 0.052), c(0.351, 0.464, 0.146, 0.039))
rownames(dados) <- c("0-19", "20-49", "50-69", "70+")
colnames(dados) <- c("Mulheres", "Homens")
dados
```

```
##       Mulheres Homens
## 0-19     0.328  0.351
## 20-49    0.461  0.464
## 50-69    0.159  0.146
## 70+      0.052  0.039
```

Plot

A função barplot é expecifica para usar em gráficos de barras horizontais e verticais. Neste caso dos dados são uma pequena tabela com duas colunas e quatro linhas. Pode-se mostrar uma barra para cada grupo (coluna) com as informações das linhas embilhadas umas sobre as outras. Neste gráfico as cores serão em

tons de cinza para gerar quatro padrões de prencimento das barras. Além disso, uma legenda a direita das barras será adicionada, bem como uma linha no eixo x na altura zero.

Argumentos:

col -Especificar cores usadas no gráfico.

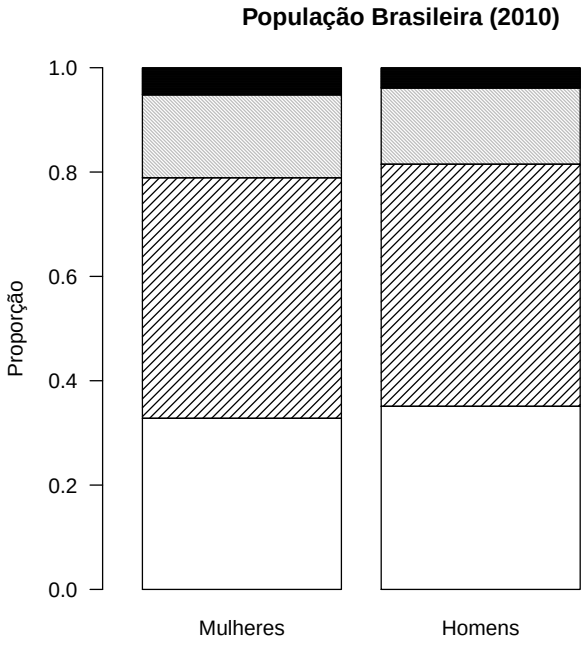
density - Especificar a densidade das linhas de sombreado para as barras. Valores em linhas por polegada.

angle - Especificar angulo da direção das linhas de sombreamento.

xlim e **ylim** - Especificar limites para os eixos x e y respectivamente.

xaxs - Especificar métodos para calcular os limites dos eixos.

```
barplot(dados,
    col = c("white", "black", "grey", "black"),
    density = c(0, 20, 50, 200),
    angle = c(0,45, 135, 0),
    ylab = "Proporção",
    las = 1,
    xaxs = "i",
    main = "População Brasileira (2010)",
    ylim = c(0, 1), xlim = c(0, 3))
```



Para adicionar legenda a opção mais fácil é especificar diretamente na função barplot. Para adicionar uma linha no limite para o eixo x na altura do zero pode-se usar a função axis. Neste último caso é importante que os limites dos eixos sejam calculados exatamente como o expecificado pelos limites.

Argumentos:

legend - Especificar nomes para os itens da legenda.

args.legend - Especificar argumentos adicionas para a legenda.

x - Parte de args.legend usado para especificar posição da legenda.

bty - Parte de args.legend usado para especificar tipo de borda da legenda.

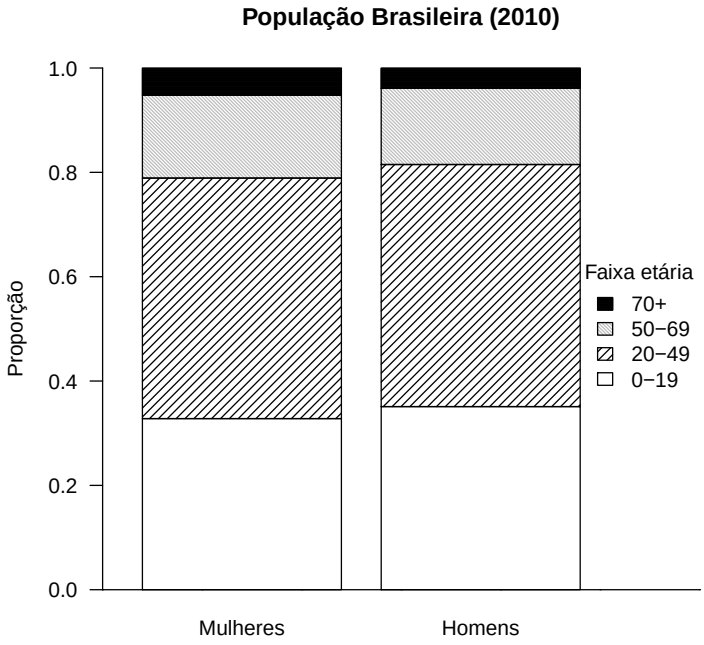
title - Parte de args.legend usado para especificar título da legenda.

side - Especificar posição do eixo.

label - Especificar nomes para os separadores do eixo.

tck - Especificar tamanho dos marcadores do eixo.

```
barplot(dados,
    col = c("white", "black", "grey", "black"),
    density = c(0, 20, 50, 200),
    angle = c(0,45, 135, 0),
    ylab = "Proporção",
    las = 1,
    xaxs = "i",
    main = "População Brasileira (2010)",
    ylim = c(0, 1), xlim = c(0, 3),
    legend = rownames(dados),
    args.legend = list(x = "right",bty = "n",
    title = "Faixa etária"))

axis(side = 1, label = F, tck = 0)
```



3.3 Gráfico de pontos com linha de regressão

Dados

Dados anscombe são um conjunto para regressoes lineares. São 4 pares de dados com variável independente e dependente com 11 observações.

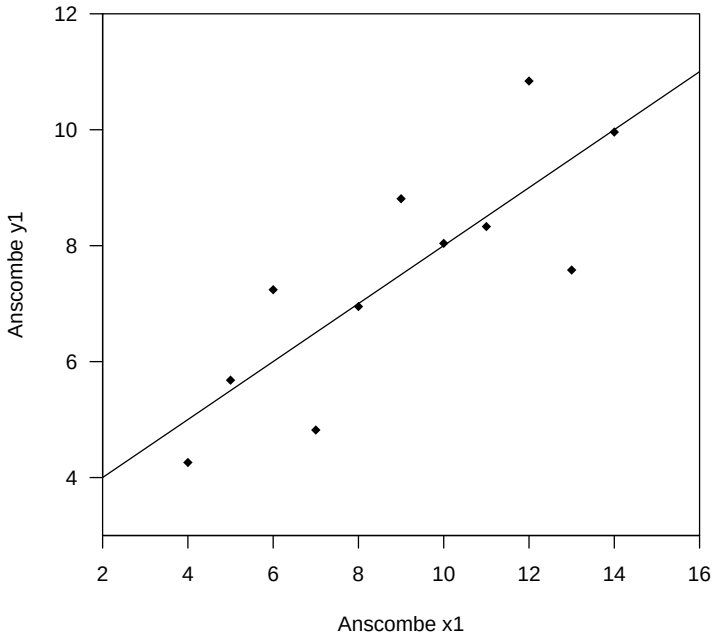
```
data(anscombe)
dados <- anscombe
str(dados)

## 'data.frame': 11 obs. of  8 variables:
##  $ x1: num  10 8 13 9 11 14 6 4 12 7 ...
```

```
##  $ x2: num  10 8 13 9 11 14 6 4 12 7 ...
##  $ x3: num  10 8 13 9 11 14 6 4 12 7 ...
##  $ x4: num  8 8 8 8 8 8 8 19 8 8 ...
##  $ y1: num  8.04 6.95 7.58 8.81 8.33 ...
##  $ y2: num  9.14 8.14 8.74 8.77 9.26 8.1 6.13 3.1 9.13 7.26 ...
##  $ y3: num  7.46 6.77 12.74 7.11 7.81 ...
##  $ y4: num  6.58 5.76 7.71 8.84 8.47 7.04 5.25 12.5 5.56 7.91 ...
```

Plot

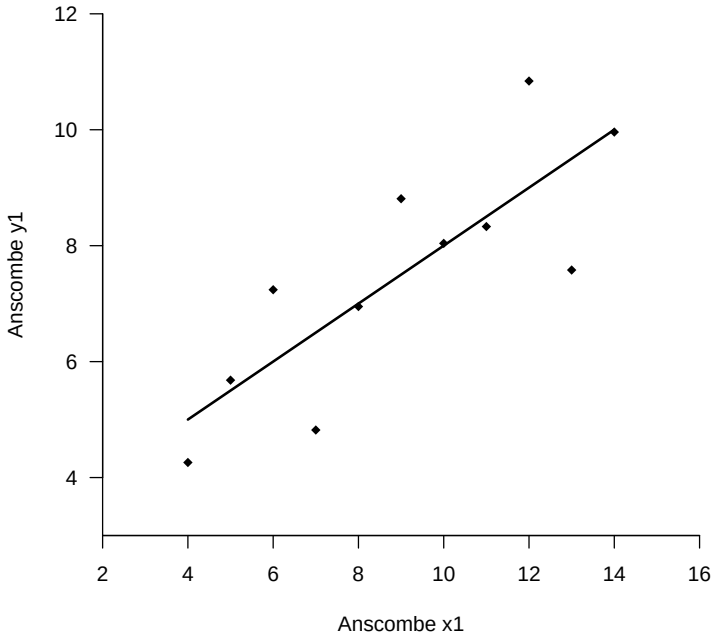
Gráfico de pontos com linha de regressão linear e o intervalo de confirança. A opção mais simples de mostrar a linha do modelo linear é usar a função abline.

```
plot(dados$x1, dados$y1,
    xlab = "Anscombe x1", ylab = "Anscombe y1",
    las = 1,
    xlim = c(2, 16),ylim = c(3, 12),
    xaxs = "i", yaxs = "i",
    pch = 18)

mod <- lm(dados$y1 ~ dados$x1)
mod # Modelo linear

##
## Call:
## lm(formula = dados$y1 ~ dados$x1)
##
## Coefficients:
## (Intercept)     dados$x1
##         3.0          0.5

abline(mod)
```



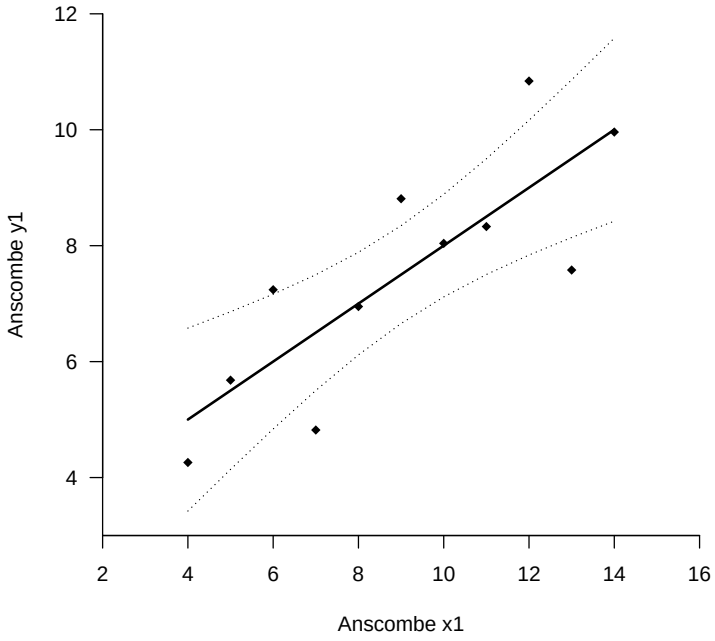
Para restringir a linha no intervalo dos dados não é possivel usar a função abline. Para fazer isso usa-se a função lines com os valores preditos pelo modelo linear. Uma exigência para a função lines funcionar corretamente é ordenar os dados na sequência de aparecem em x. A função lines simplesmente conecta os pontos na sequencia que são fornecidos, se a sequencia não está em ordem, crescente ou descresente, a linha não ficará como o esperado. A função order é usada para ordenar os valores.

```
fitted <- predict(mod, interval = "confidence") # Valores preditos
str(fitted)

##  num [1:11, 1:3] 8 7 9.5 7.5 8.5 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:11] "1" "2" "3" "4" ...
##   ..$ : chr [1:3] "fit" "lwr" "upr"

plot(dados$x1, dados$y1,
    xlab = "Anscombe x1", ylab = "Anscombe y1",
    las = 1,
    xlim = c(2, 16),ylim = c(3, 12),
    xaxs = "i", yaxs = "i",
    pch = 18,
    bty = "l")

lines(dados$x1[order(dados$x1)], predict(mod)[order(dados$x1)],
    lty = 1,
    lwd = 2)
```



Para mostrar os intevalos de confiança da regressão também usa-se a mesma função lines. Neste caso se os valores de x não estiverem em sequência crescente ou descresente a linha não será mostrada corretamente. A função curve (ver ?curve) pode produzir um linhas semelhante, em alguns casos com melhores resultados, mas seu funcionamento depende de determinar uma função para a curva que será mostrada.

```
plot(dados$x1, dados$y1,
    xlab = "Anscombe x1", ylab = "Anscombe y1",
    las = 1,
    xlim = c(2, 16),ylim = c(3, 12),
    xaxs = "i", yaxs = "i",
    pch = 18,
    bty = "l")

lines(dados$x1[order(dados$x1)], predict(mod)[order(dados$x1)],
    lty = 1,
    lwd = 2)

lines(dados$x1[order(dados$x1)], fitted[, "lwr"][order(dados$x1)],
    lty = 3)

lines(dados$x1[order(dados$x1)], fitted[, "upr"][order(dados$x1)],
    lty = 3)
```



3.4 Gráfico pontos com ajustes não lineares

Dados

Conjunto de dados simulados sem relação linear, sendo valores entre 1 e 100 com variável independente e como variável dependende os dados são uma função do logaritmo da variável dependente mais um componente aleátorio. Quatro modelos são gerados, cada um com uma relação entre as variáveis. Os coeficientes dos modelos são organizados e servirão para construir as funções dos modelos e para mostrar na legenda.

```
set.seed(1)
dados <- data.frame(dx = 1:100, dy = 0.4 + (4.8 * log(1:100)) + rnorm(100, 0, 1))
str(dados)
```

```
## 'data.frame': 100 obs. of  2 variables:
##  $ dx: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ dy: num  -0.226 3.911 4.838 8.649 8.455 ...
```

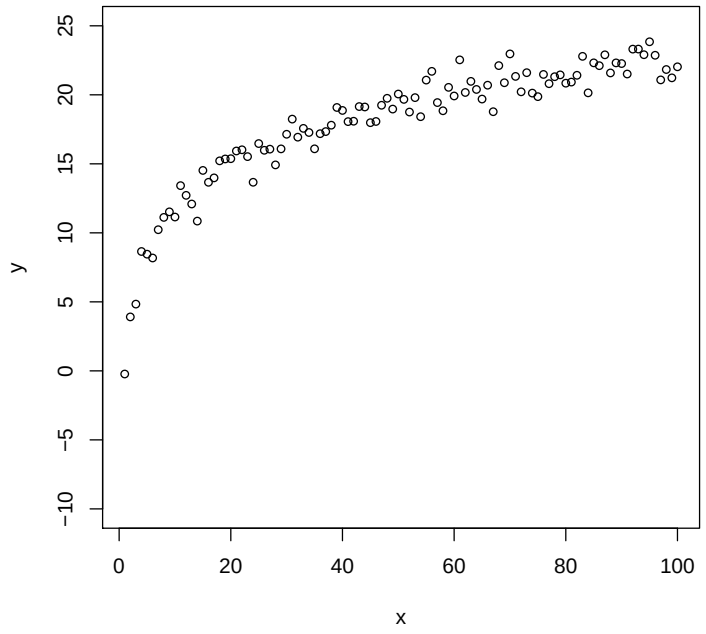
```
mod1 <- lm(dy ~ 1, data = dados)  # Média
mod2 <- lm(dy ~ dx, data = dados)  # Linear
mod3 <- nls(dy ~ a + b * log(dx), data = dados, start = list(a = 0, b = 0))  # Logaritmo
mod4 <- nls(dy ~ a + b/dx, data = dados, start = list(a = 0, b = 0))  # Assintótico

coeficientes <- data.frame(m1 = paste("y = ", round(coefficients(mod1)[1], 2), sep = ""),
    m2 = paste("y = ", round(coefficients(mod2)[1], 2),
        " + (", round(coefficients(mod2)[2], 2), "*x)", sep = ""),
    m3 = paste("y = ", round(coefficients(mod3)[1], 2),
        " + (", round(coefficients(mod3)[2],2), "*log(x))", sep = ""),
    m4 = paste("y = ", round(coefficients(mod4)[1], 2),
        " + (",round(coefficients(mod4)[2], 2), "/x)", sep = ""))
coeficientes  # Coeficientes para cada modelo
```

```
##          m1                    m2                      m3                      m4
## 1 y = 17.97 y = 11.04 + (0.14*x) y = 0.5 + (4.8*log(x)) y = 19.51 + (-29.64/x)
```

Plot

Construir um gráfico de pontos e mostrar uma linha com a função de cada um dos modelos, lineares ou não. Além disso, adicionar a legenda com os parâmetros estimados.

```
plot(dy ~ dx, data = dados,
    ylim = c(-10, 25),
    pch = 21,
    cex = 0.8,
    bty = "o",
    ylab = "y", xlab = "x")
```



Para mostrar a curva com cada um dos modelos, pode-se usar a função lines ou a função curve. No caso da função curve deve-se construir uma função para cada uma das linhas. A função é contruida com base nos parâmetros estimados para cada um dos modelos, para isso basta usar function(x) e adiconar os parâmetros de cada modelo.

Argumentos:

from - Valor inicial do limite onde a função é mostrada

to - Valor final do limite onde a função é mostrada

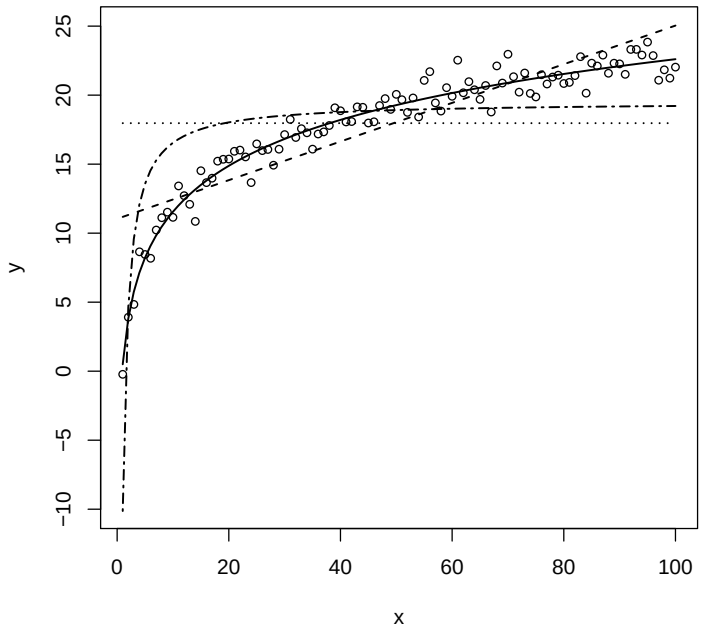
add - Argumento lógico para especificar se curvas devem ser adicionadas ao gráfico existente ou não. Padrão = FALSE.

lty - Tipo de linha para a curva.

lwd - Espessura da linha na curva.

```
fmod1 <- function(x) 17.97 + (0 * x)
fmod2 <- function(x) 11.04 + (0.14 * x)
fmod3 <- function(x) 0.5 + (4.8 * log(x))
```

```r
fmod4 <- function(x) 19.51 + (-29.64/x)

plot(dy ~ dx, data = dados,
     ylim = c(-10, 25),
     pch = 21,
     cex = 0.8,
     bty = "o",
     ylab = "y", xlab = "x")

curve(fmod1,
     from = 1,
     to = 100,
     add = TRUE,
     lty = 3,
     lwd = 1.5)

curve(fmod2,
     from = 1,
     to = 100,
     add = TRUE,
     lty = 2,
     lwd = 1.5)

curve(fmod3,
     from = 1,
     to = 100,
     add = TRUE,
     lty = 1,
     lwd = 1.5)

curve(fmod4,
     from = 1,
     to = 100,
     add = TRUE,
     lty = 6,
     lwd = 1.5)
```



Adicionar a legenda com os parâmetros estimados. Os parâmetros estão organizados dentro do objeto coeficientes.

Argumentos:

legend - Especificar nomes para os itens da legenda.

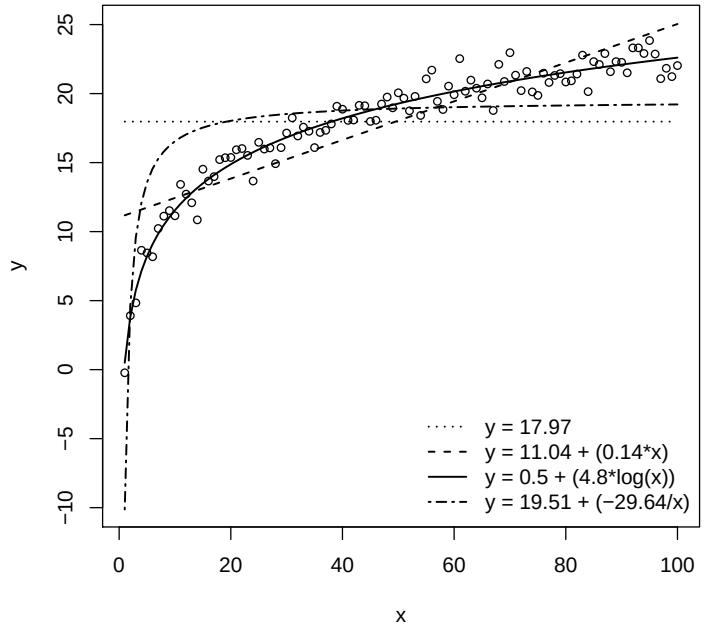
bty - Especificar tipo de borda da legenda.

lty - Tipo de linha para cada curva.

lwd - Espessura da linha na curva.

```
plot(dy ~ dx, data = dados,
     ylim = c(-10, 25),
     pch = 21,
     cex = 0.8,
     bty = "o",
     ylab = "y", xlab = "x")

curve(fmod1,
     from = 1,
     to = 100,
     add = TRUE,
     lty = 3,
     lwd = 1.5)

curve(fmod2,
     from = 1,
     to = 100,
     add = TRUE,
     lty = 2,
     lwd = 1.5)

curve(fmod3,
```

```
    from = 1,
    to = 100,
    add = TRUE,
    lty = 1,
    lwd = 1.5)

curve(fmod4,
    from = 1,
    to = 100,
    add = TRUE,
    lty = 6,
    lwd = 1.5)

legend("bottomright",
    legend = c(paste(coeficientes$m1), paste(coeficientes$m2),
        paste(coeficientes$m3),paste(coeficientes$m4)),
    bty = "n",
    lty = c(3, 2, 1, 6),
    lwd = 1.5)
```



3.5 Diagrama de dispersão

Dados

Dados sobre criminalidade dos Estados Unidos da America em 1973, descrito por quatro variáveis. Pode-se aplicar uma Análise de Componentes Principais (PCA) e usar os scores para contruir o gráfico apenas com os dois primeiros eixos.

```r
data(USArrests)
str(USArrests)
```

```
## 'data.frame': 50 obs. of  4 variables:
##  $ Murder  : num  13.2 10 8.1 8.8 9 7.9 3.3 5.9 15.4 17.4 ...
##  $ Assault : int  236 263 294 190 276 204 110 238 335 211 ...
##  $ UrbanPop: int  58 48 80 50 91 78 77 72 80 60 ...
##  $ Rape    : num  21.2 44.5 31 19.5 40.6 38.7 11.1 15.8 31.9 25.8 ...
```

```r
require(FactoMineR)

Rpca <- PCA(USArrests, graph = FALSE)  # PCA
str(Rpca$eig)  # Autovalores
```

```
## 'data.frame': 4 obs. of  3 variables:
##  $ eigenvalue                      : num  2.48 0.99 0.357 0.173
##  $ percentage of variance          : num  62.01 24.74 8.91 4.34
##  $ cumulative percentage of variance: num  62 86.8 95.7 100
```

```r
str(Rpca$var$coord)  # Correlação com os eixos
```

```
##  num [1:4, 1:4] 0.844 0.918 0.438 0.856 -0.416 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:4] "Murder" "Assault" "UrbanPop" "Rape"
##   ..$ : chr [1:4] "Dim.1" "Dim.2" "Dim.3" "Dim.4"
```

```r
str(Rpca$ind$coord)  # Scores das unidades amostrais
```

```
##  num [1:50, 1:4] 0.986 1.95 1.763 -0.141 2.524 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:50] "Alabama" "Alaska" "Arizona" "Arkansas" ...
##   ..$ : chr [1:4] "Dim.1" "Dim.2" "Dim.3" "Dim.4"
```

```r
dados_unidades <- Rpca$ind$coord[, 1:2]
str(dados_unidades) # Scores das unidades amostrais para os dois primeiros eixos
```

```
##  num [1:50, 1:2] 0.986 1.95 1.763 -0.141 2.524 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:50] "Alabama" "Alaska" "Arizona" "Arkansas" ...
##   ..$ : chr [1:2] "Dim.1" "Dim.2"
```

```r
dados_variaveis <- Rpca$var$coord[, 1:2]
str(dados_variaveis) # Correlação das variáveis para os dois primeiros eixos
```

```
##  num [1:4, 1:2] 0.844 0.918 0.438 0.856 -0.416 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:4] "Murder" "Assault" "UrbanPop" "Rape"
##   ..$ : chr [1:2] "Dim.1" "Dim.2"
```

Plot

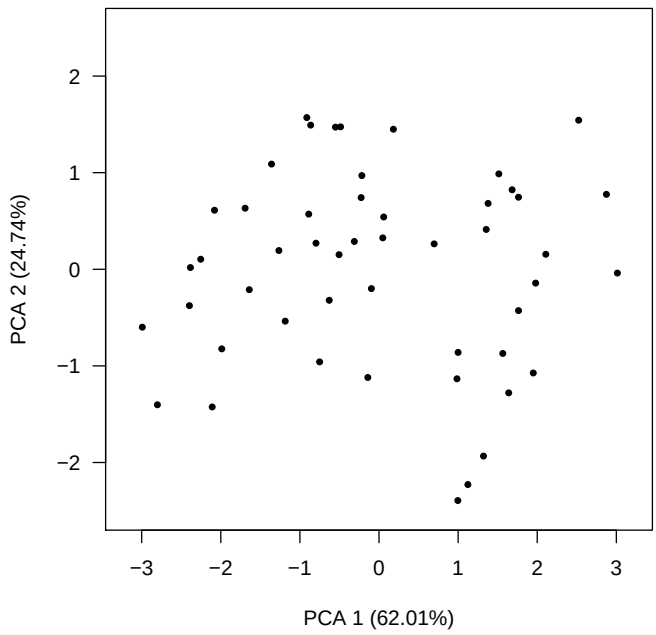
Para construir o diagrama de dispersão e mostrar simultaneamente os dados das unidades amostrais e a correlação com as variáveis o gráfico precisa ter quatro eixos, com duas escalas diferentes. A primeira coisa que deve ser feita é alterar as margens padrão do gráfico. O argumento mar deve ser alterado antes de começar a construir o gráfico na função par.

Argumentos:

mar - Especificar margens para o gráfico. O argumento é do tipo vector que indica o número de linhas de margem para os quatro lados na forma c(margem inferior, à esquerda, superior, margem à direita). O padrão é c (5.1, 4.1, 4.1, 2.1).

pch - Tipo de símbolo para os pontos.

```
par(mar = c(5.1, 4.1, 4.1, 4.1))

plot(dados_unidades,
    pch = 19,
    cex = 0.6,
    ylab = "PCA 2 (24.74%)", xlab = "PCA 1 (62.01%)",
    xlim = c(-3.2, 3.2), ylim = c(-2.5, 2.5),
    las = 1)
```



A segunda parte do gráfico consiste em sobrepor sobre o mesmo gráfico o segundo conjunto de dados. A função par é usada novamente com o argumento new = TRUE, isso faz com que o próximo gráfico seja construido sobreposto ao gráfico antigo. Eixos e nomes dos eixos não devem ser mostrados, eles deverão ser adicionados pela função axis.

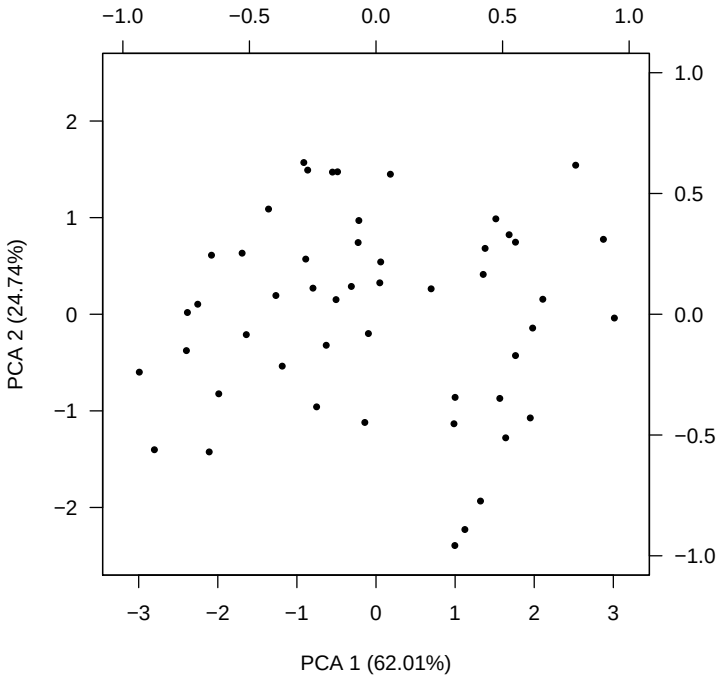
Argumentos:

new - Argumento lógico para especificar ser próximo gráfico deve ser construido sem apagar o gráfico antigo.

side - Especificar lado que o eixo deve ser mostrado. Valores aceitos 1 (eixo inferior), 2 (eixo à esquerda), 3 (eixo superior) e 4 (eixo à direita).

```
par(mar = c(5.1, 4.1, 4.1, 4.1))

plot(dados_unidades,
    pch = 19,
    cex = 0.6,
    ylab = "PCA 2 (24.74%)", xlab = "PCA 1 (62.01%)",
    xlim = c(-3.2, 3.2), ylim = c(-2.5, 2.5),
```

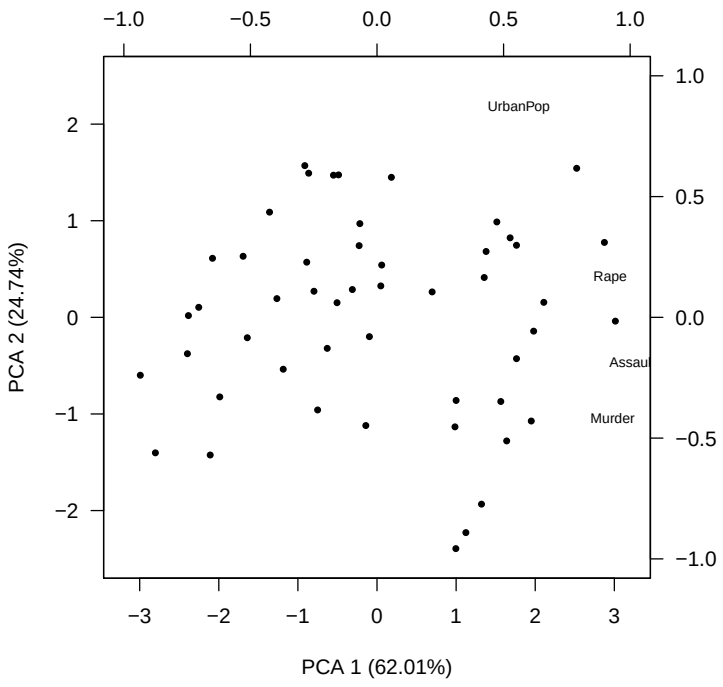
```
    las = 1)

par(new = TRUE)

plot(dados_variaveis,
    xlab = "", ylab = "",
    xaxt = "n", yaxt = "n",
    type = "n",
    ylim = c(-1,1), xlim = c(-1, 1))

axis(side = 4,
    las = 1)

axis(side = 3)
```



O segundo conjuntos de pontos será mostrado na forma de texto.
Argumentos:
labels - Especificar texto a ser adicionado em cada coordenada.
cex - Especificar tamanho da fonte do texto.
adj - Especificar se o texto deve se mostrado centrado exatamente da coordenada ou adjacente a coordenada. Argumento do tipo c(x, y), onde os valores representam o ajuste nos eixos x e y respectivamente. Se adj for 0 os texto é mostrado justificado a esquerda da coordenada.

```
par(mar = c(5.1, 4.1, 4.1, 4.1))

plot(dados_unidades,
    pch = 19,
    cex = 0.6,
    ylab = "PCA 2 (24.74%)", xlab = "PCA 1 (62.01%)",
    xlim = c(-3.2, 3.2), ylim = c(-2.5, 2.5),
```

```
    las = 1)

par(new = TRUE)

plot(dados_variaveis,
    xlab = "", ylab = "",
    xaxt = "n", yaxt = "n",
    type = "n",
    ylim = c(-1,1), xlim = c(-1, 1))

axis(side = 4,
    las = 1)

axis(side = 3)

text(dados_variaveis,
    labels = rownames(dados_variaveis),
    cex = 0.7,
    adj = 0)
```



Para finalizar o gráfico adiciona-se setas que partem da origem até os valores de cada uma das coordenadas de texto e uma linha simples no valor 0 para ambos os eixos.

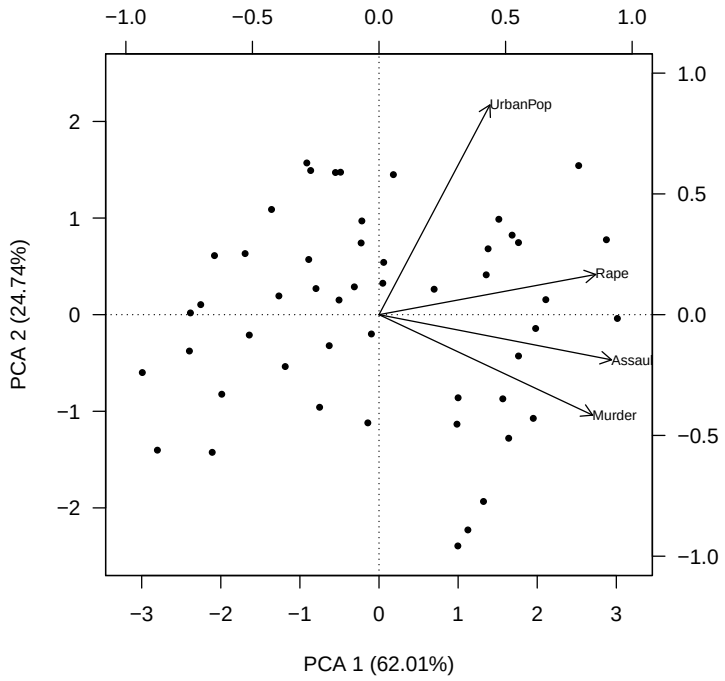
Argumentos:

lenght - Comprimento da ponta da seta, em polegadas.

h e **v** - Posição para adicionar linha na horizontal e vertical respectivamente.

```
par(mar = c(5.1, 4.1, 4.1, 4.1))

plot(dados_unidades,
    pch = 19,
```

```
    cex = 0.6,
    ylab = "PCA 2 (24.74%)", xlab = "PCA 1 (62.01%)",
    xlim = c(-3.2, 3.2), ylim = c(-2.5, 2.5),
    las = 1)

par(new = TRUE)

plot(dados_variaveis,
    xlab = "", ylab = "",
    xaxt = "n", yaxt = "n",
    type = "n",
    ylim = c(-1,1), xlim = c(-1, 1))

axis(side = 4,
    las = 1)

axis(side = 3)

text(dados_variaveis,
    labels = rownames(dados_variaveis),
    cex = 0.7,
    adj = 0)

arrows(x0 = 0, y = 0,
    x1 = dados_variaveis[, 1], y1 = dados_variaveis[, 2],
    length = 0.1)

abline(h = 0,
    v = 0,
    lty = 3)

par(resetPar) # Restaurar parâmetros originais.
```



3.6 Boxplot

Dados

Dados de experimento sobre o rendimento (peso seco de plantas) obtidos sob controle e dois tratamento. Uma ANOVA seguida por um teste de Tukey foram realizados para verificar se existem diferenças significativas entre os tratamentos.

```
data(PlantGrowth)
dados <- PlantGrowth
str(dados)

## 'data.frame': 30 obs. of  2 variables:
##  $ weight: num  4.17 5.58 5.18 6.11 4.5 4.61 5.17 4.53 5.33 5.14 ...
##  $ group : Factor w/ 3 levels "ctrl","trt1",..: 1 1 1 1 1 1 1 1 1 1 ...

mod <- lm(weight ~ group, data = dados)  # ANOVA
summary.aov(mod)

##             Df Sum Sq Mean Sq F value Pr(>F)
## group        2   3.77   1.883    4.85  0.016 *
## Residuals   27  10.49   0.389
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

TukeyHSD(aov(weight ~ group, data = dados))  # Tukey

##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = weight ~ group, data = dados)
```

```
##
## $group
##              diff     lwr    upr  p adj
## trt1-ctrl -0.371 -1.0622 0.3202 0.3909
## trt2-ctrl  0.494 -0.1972 1.1852 0.1980
## trt2-trt1  0.865  0.1738 1.5562 0.0120
```

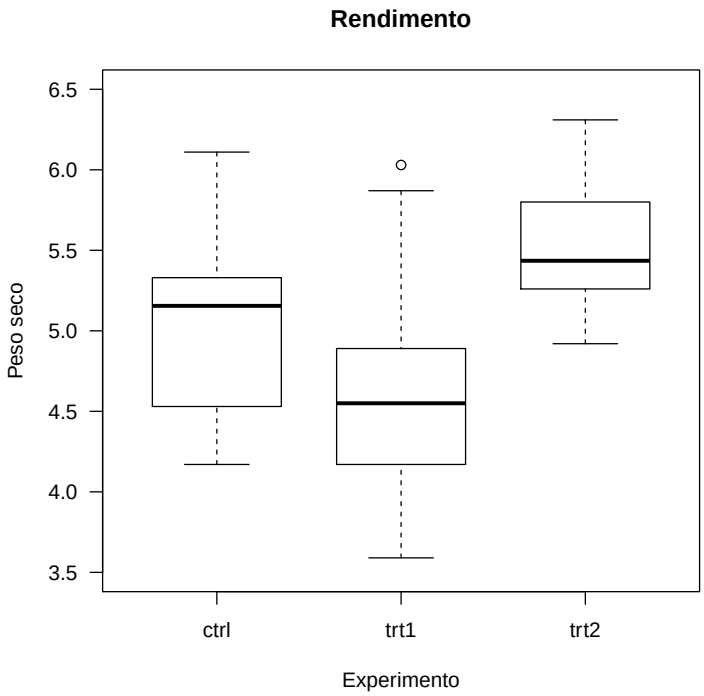
Plot

Construir um gráfico do tipo boxplot permite visualizar o efeito dos tratamentos. Neste caso é usado a notação de fórmula para as variáveis.

Argumento:

boxwex - Especificar largura das caixas.

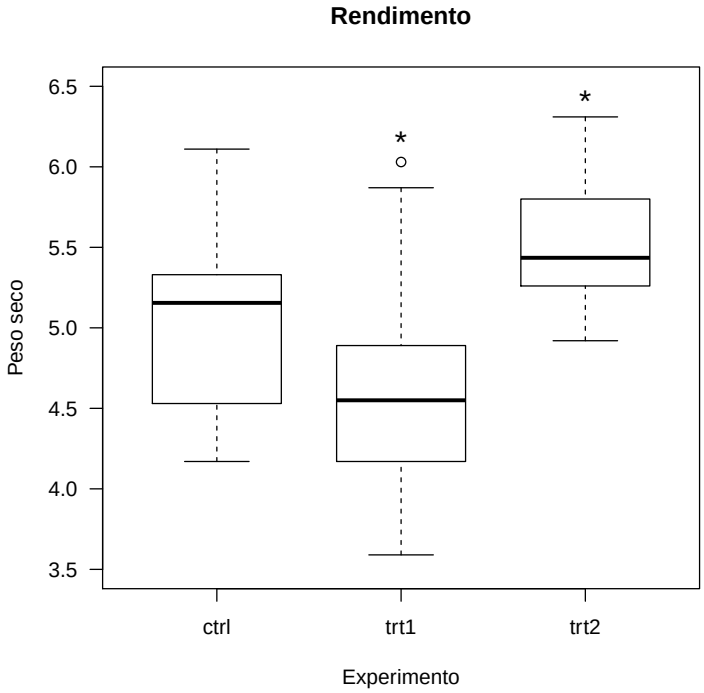
```
boxplot(weight ~ group,
    data = dados,
    ylab = "Peso seco", xlab = "Experimento",
    main = "Rendimento",
    boxwex = 0.7,
    las = 1,
    ylim = c(3.5, 6.5))
```



Pode-se usar a função text para destacar no gráfico as diferenças significativas entre os tratamentos.

```
boxplot(weight ~ group,
    data = dados,
    ylab = "Peso seco", xlab = "Experimento",
    main = "Rendimento",
    boxwex = 0.7,
    las = 1,
```

```
    ylim = c(3.5, 6.5))

text(c(2, 3), c(6.2, 6.45),
    labels = "*",
    cex = 1.5)
```



Para mudar a ordem dos tratamentos no gráfico precisa-se alterar a ordem dos factores nos dados. Além disso, pode-se inverter as cores das caixas e trocar o símbolo dos outliers.

Argumentos:

col - Cor das caixas.

border - Cor das bordas das caixas.

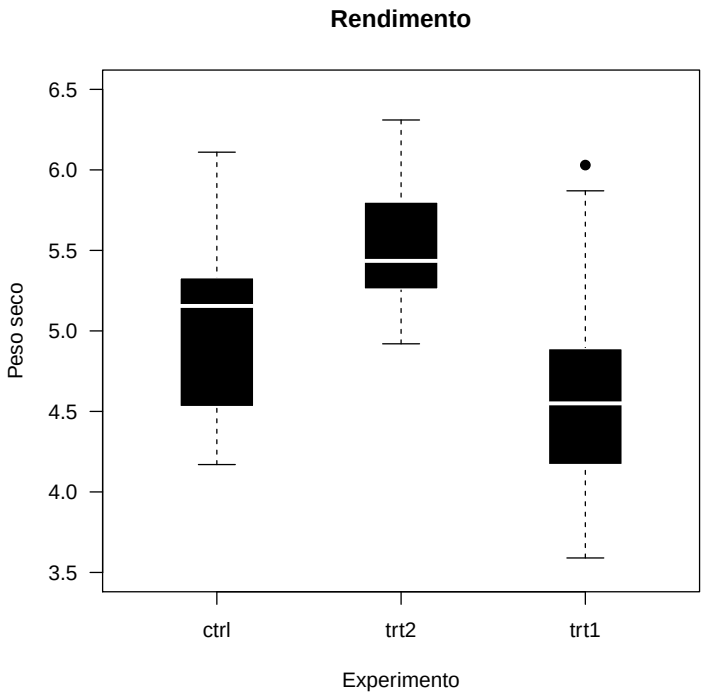
whiskcol - Cor das linhas.

staplecol - Cor das barras limites.

outpch - Tipo de símbolo para os outliers.

```
dados$group <- factor(dados$group, levels = c("ctrl", "trt2", "trt1"))
str(dados)

## 'data.frame': 30 obs. of  2 variables:
##  $ weight: num  4.17 5.58 5.18 6.11 4.5 4.61 5.17 4.53 5.33 5.14 ...
##  $ group : Factor w/ 3 levels "ctrl","trt2",..: 1 1 1 1 1 1 1 1 1 1 ...

boxplot(weight ~ group,
    data = dados,
    ylab = "Peso seco", xlab = "Experimento",
    main = "Rendimento",
    las = 1,
    ylim = c(3.5, 6.5),
    col = "black",
    border = "white",
    whiskcol = "black",
```

```
    staplecol = "black",
    outcol = "black",
    outpch = 19,
    boxwex = 0.4)
```



Para finalizar pode-se alterar os nomes para dos tratamentos e alterar o range das barras do boxplot. Por padrão o range das barras do boxplot estendem-se até 1.5 desvios quantílicos, que é a medido da diferença entre o primeiro e o terceiro quatil da variável.

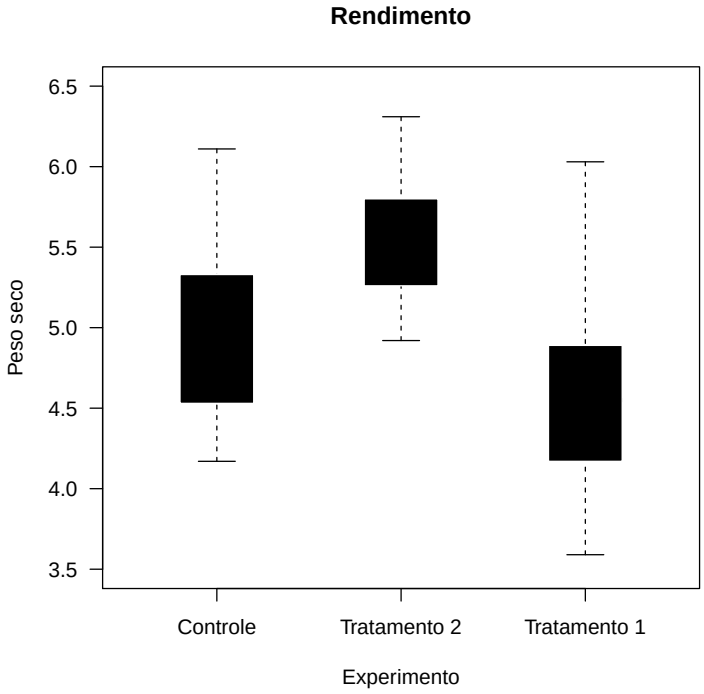
Argumentos:

names - Alterar nome dos níveis.

medcol - Alterar cores da linha do meio.

range - Determinar o range das barras. Padrão é 1.5, se for 0 as barras são mostradas até o limites dos dados.

```
boxplot(weight ~ group,
    data = dados,
    ylab = "Peso seco", xlab = "Experimento",
    main = "Rendimento",
    las = 1,
    ylim = c(3.5, 6.5),
    col = "black",
    border = "white",
    whiskcol = "black",
    staplecol = "black",
    outcol = "black",
    medcol = "black",
    outpch = 19,
    range = 0,
    boxwex = 0.4,
    names = c("Controle","Tratamento 2", "Tratamento 1"))
```



3.7 Gráfico de interação

Dados

Dados iron disponíveis no pacote AICcmodavg descrevem o teor de ferro de alimentos cozidos em diferentes tipos de panelas. Nos dados são dois factores e uma varíavel dependente. Pode-se calcular a média para cada nível dos dois factores combinados.

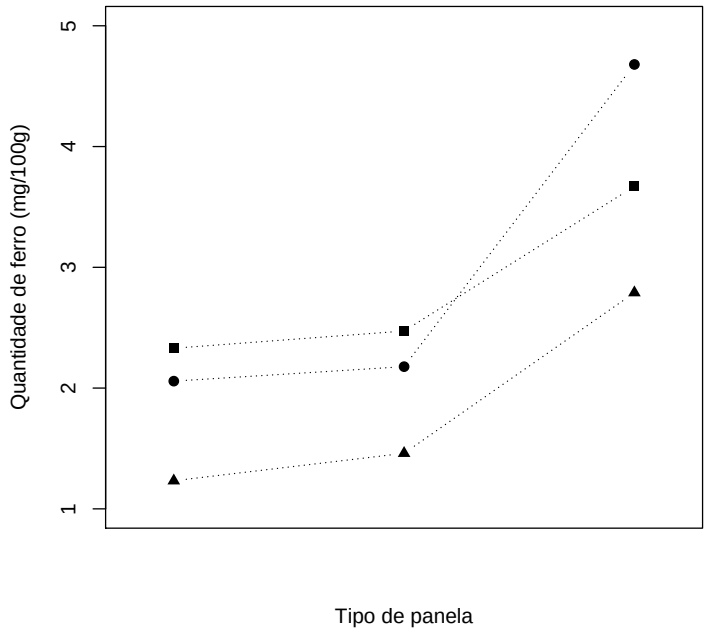
```
require(AICcmodavg)
data(iron)
str(iron)

## 'data.frame': 36 obs. of  3 variables:
##  $ Pot : Factor w/ 3 levels "aluminium","clay",..: 1 1 1 1 2 2 2 2 3 3 ...
##  $ Food: Factor w/ 3 levels "legumes","meat",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ Iron: num  1.77 2.36 1.96 2.14 2.27 1.28 2.48 2.68 5.27 5.17 ...

dados <- tapply(iron[, 3], list(iron[, 1], iron[, 2]), mean)
dados

##           legumes  meat vegetables
## aluminium   2.330 2.058      1.232
## clay        2.473 2.178      1.460
## iron        3.670 4.680      2.790
```

Plot

O gráfico de interação pode mostrar a influência dos dois factores sobre a variável dependente. Pode-se usar a função matplot, que funciona muito bem para este tipo de dados, mas não é tão simples. Alternativamente pose-se usar a função interaction.plot, que já calcula a média para cada tratamento, e construi

o gráfico, mas esta não permite tantas personalizações. Primeiramente define-se os limites para o eixo x. Neste caso, os dados são três colunas e três linhas, os limites podem ser definidos com valores um pouco abaixo de um e um pouco acima de três, pois cada nível será mostrado sobre o número inteiro de um a três. Além disso não é mostrado nenhuma informação sobre o eixo x.

```
matplot(dados,
    type = "b",
    xlim = c(0.8, 3.2), ylim = c(1, 5),
    lty = 3,
    lwd = 1,
    pch = c(15,19, 17),
    col = "black",
    xaxt = "n",
    xlab = "Tipo de panela", ylab = "Quantidade de ferro (mg/100g)")
```



Para adicionar o eixo x personalizado, basta definir os marcadores para o eixo e os nomes conforme o tipo do nome da variável. É importante verificar qual dos nívels é mostrado no eixo x e qual dos nívels é mostrado nas linhas.

Argumentos:

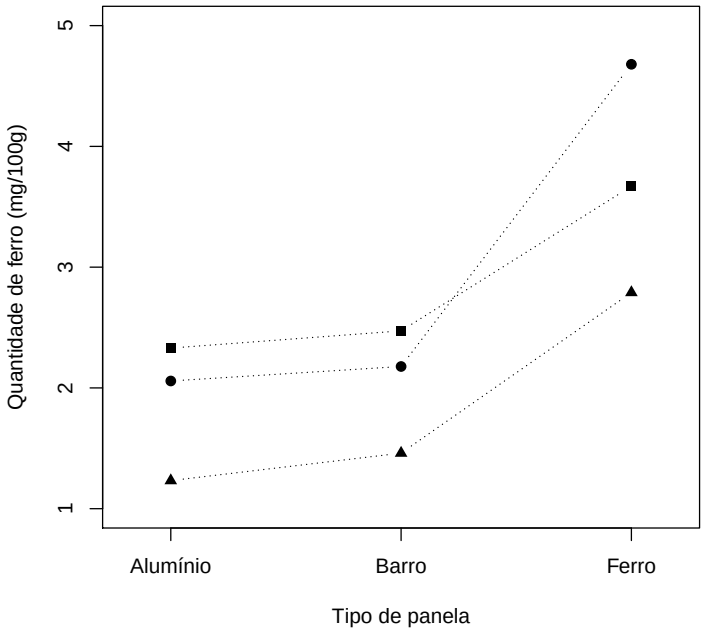
side - Lado para mostrar o eixo.

at - Localização dos marcadores do eixo. Um vector com todos os separadores.

labels - Texto para adicionar em cada posição do marcador.

```
matplot(dados,
    type = "b",
    xlim = c(0.8, 3.2), ylim = c(1, 5),
    lty = 3,
    lwd = 1,
    pch = c(15,19, 17),
    col = "black",
```

```
    xaxt = "n",
    xlab = "Tipo de panela", ylab = "Quantidade de ferro (mg/100g)")

axis(side = 1,
    at = 1:3,
    labels = c("Alumínio", "Barro", "Ferro"))
```



Para finalizar só resta adicionar a legenda ao gráfico.

Argumentos:

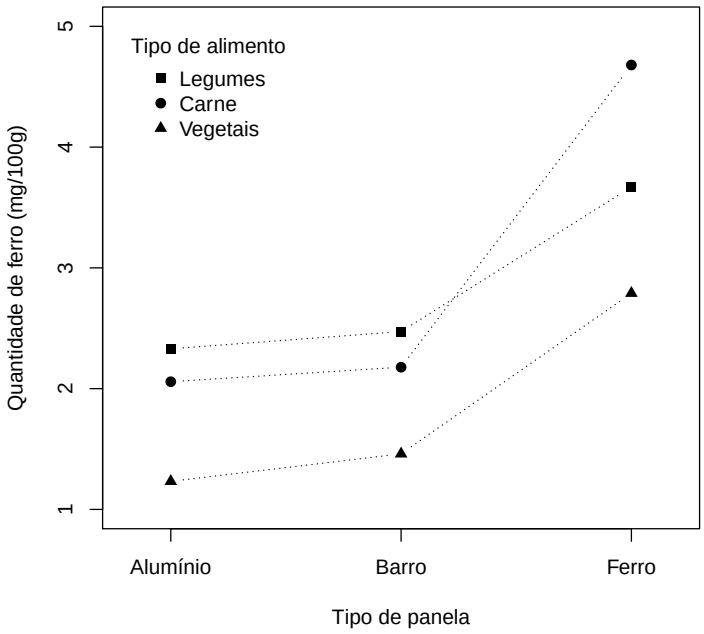
pch - Tipo de marcado para cada nível. Nesta caso também é importante verificar a ordem de cada nível nos dados.

inset - Deslocamento da inserção da legenda, tendo como referência a palavra-chave da localização.

title - Título para a legenda.

```
matplot(dados,
    type = "b",
    xlim = c(0.8, 3.2), ylim = c(1, 5),
    lty = 3,
    lwd = 1,
    pch = c(15,19, 17),
    col = "black",
    xaxt = "n",
    xlab = "Tipo de panela", ylab = "Quantidade de ferro (mg/100g)")

axis(side = 1,
    at = 1:3,
    labels = c("Alumínio", "Barro", "Ferro"))

legend("topleft",
    legend = c("Legumes", "Carne", "Vegetais"),
```

```
    pch = c(15, 19, 17),
    bty = "n",
    inset = 0.04,
    title = "Tipo de alimento")
```



3.8 Gráfico de pontos com barras de erros

Dados

As contagens de insetos em unidades experimentais agrícolas tratados com diferentes inseticidas. Pode-se calcular a média e o desvio padrão para cada inseticida.

```
data(InsectSprays)
str(InsectSprays)

## 'data.frame': 72 obs. of  2 variables:
##  $ count: num  10 7 20 14 14 12 10 23 17 20 ...
##  $ spray: Factor w/ 6 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...

require(plotrix)
dados <- tapply(InsectSprays$count, InsectSprays$spray, mean)
dados

##      A      B      C      D      E      F
## 14.500 15.333  2.083  4.917  3.500 16.667

dados_sd <- tapply(InsectSprays$count, InsectSprays$spray, sd)
dados_sd

##     A     B     C     D     E     F
## 4.719 4.271 1.975 2.503 1.732 6.213
```

Plot

Pode-se construir um gráfico de pontos representando a média de cada tratamento e linhas representando os desvios padrão. Neste caso os valores do eixo x são categoriocos, então oculta-se valores do eixo e define-se os alcance dos eixos. Os nomes dos inseticidas são adicionados em um segundo passo pela função axis.

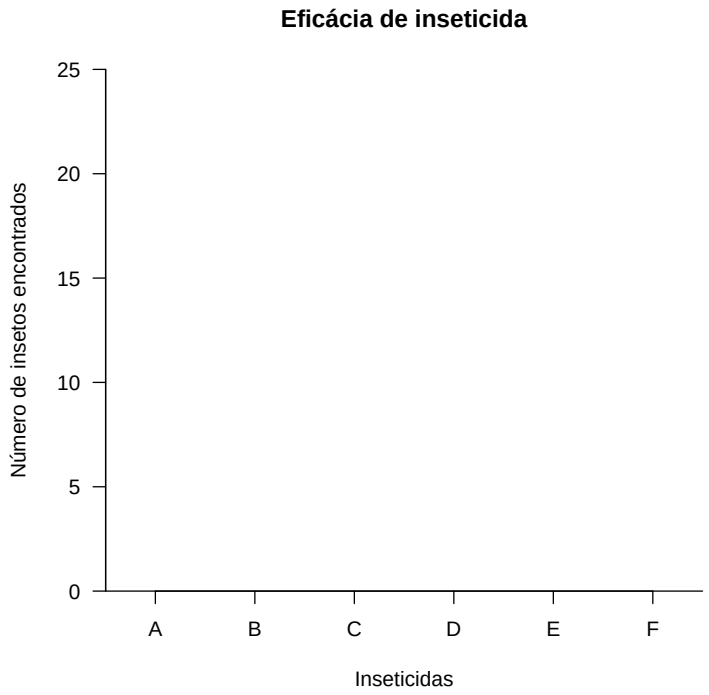
Argumentos:

side - Lado para mostrar o eixo.

at - Localização dos marcadores do eixo. Um vector com todos os separadores.

labels - Texto para adicionar em cada posição do marcador.

```
plot(1:6, dados,
    type = "n",
    xaxt = "n",
    ylim = c(0, 25), xlim = c(0.5, 6.5),
    bty = "l",
    xaxs = "i",yaxs = "i",
    ylab = "Número de insetos encontrados", xlab = "Inseticidas",
    las = 1,
    main = "Eficácia de inseticida")

axis(side = 1,
    at = 1:6,
    labels = rownames(dados))
```



Para mostrar as barras de erro a função plotCI do pacote plotrix é fácil de usar..

Argumentos:

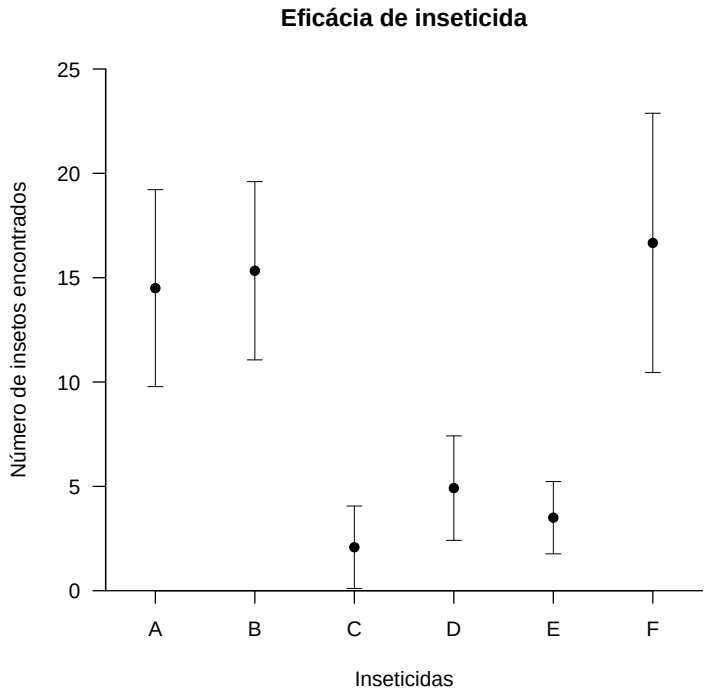
x e **y** - Coordenadas para o ínicio da barra de erro.

uiw e **liw** - Comprimento da linha de erro para a parte superior e inferior.

add - Argumento lógico para especificar se linhas de erro devem ser adicionadas ao gráfico existente ou não. Padrão é FALSE.

pch - Tipo de símbolo usado no centro da barra de erro. Se for NA não é mostrado nenhum símbolo.

```
plot(1:6, dados,
    type = "n",
    xaxt = "n",
    ylim = c(0, 25), xlim = c(0.5, 6.5),
    bty = "l",
    xaxs = "i",yaxs = "i",
    ylab = "Número de insetos encontrados", xlab = "Inseticidas",
    las = 1,
    main = "Eficácia de inseticida")

axis(side = 1,
    at = 1:6,
    labels = rownames(dados))

plotCI(1:6, dados,
    uiw = dados_sd,
    add = TRUE,
    pch = 19,
    lwd = 0.5)
```



3.9 Gráfico de linhas com duas escalas

Dados

Dados airquality descrevem a qualidade do ar de New York em alguns meses de 1973.

```
data(airquality)
dados <- airquality
```

```r
str(dados)
```

```
## 'data.frame': 153 obs. of  6 variables:
##  $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
##  $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
##  $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
##  $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
##  $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
```

Plot

Para construir um gráfico de linhas representado duas series temporais pode-se adicionar duas escalas no eixo y. Primeiramente deve-se redefinir as margens da figura. Depois mostrar a primeira série ocultando os valores e marcadores do eixo x.

Argumentos:

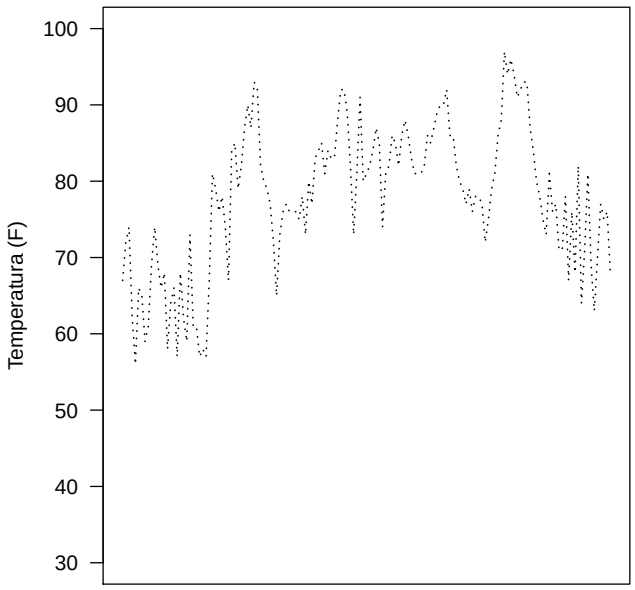
mar - Especificar margens para o gráfico. O argumento é do tipo vector que indica o número de linhas de margem para os quatro lados na forma c(margem inferior, à esquerda, superior, margem à direita). O padrão é c (5.1, 4.1, 4.1, 2.1).

type - Altera o tipo de gráfico que será construido. Vários valores são aceito, os principais são "p"(gráfico de pontos), "l"(gráfico de linhas), "b"(gráficos com pontos e linhas).

lty - Especificar tipo de linha.

lwd - Especificar espessura da linha.

```r
par(mar = c(4, 4, 3, 5), las = 1)

plot(dados$Temp,
    type = "l",
    lty = 3,
    ylim = c(30, 100),
    lwd = 1.4,
    xaxt = "n",
    ylab = "Temperatura (F)",xlab = "")
```



Os intevalos de marcação e o texto dos valores no eixo x são estabelecidos com base na divisão mensal, transformando o eixo contínuo em categórico.

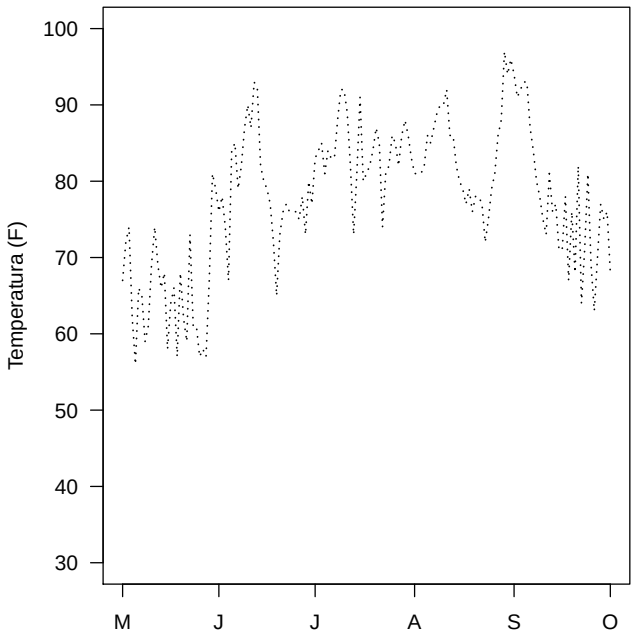
Argumentos:

side - Posição para o eixo.

at - Especificar posição dos marcadores do eixo.

label - Especificar nome para os marcadores do eixo.

```
plot(dados$Temp,
     type = "l",
     lty = 3,
     ylim = c(30, 100),
     lwd = 1.4,
     xaxt = "n",
     ylab = "Temperatura (F)",xlab = "")

axis(side = 1,
     at = c(1, cumsum(table(dados[, 5]))),
     label = c("M", "J", "J", "A", "S", "O"))
```

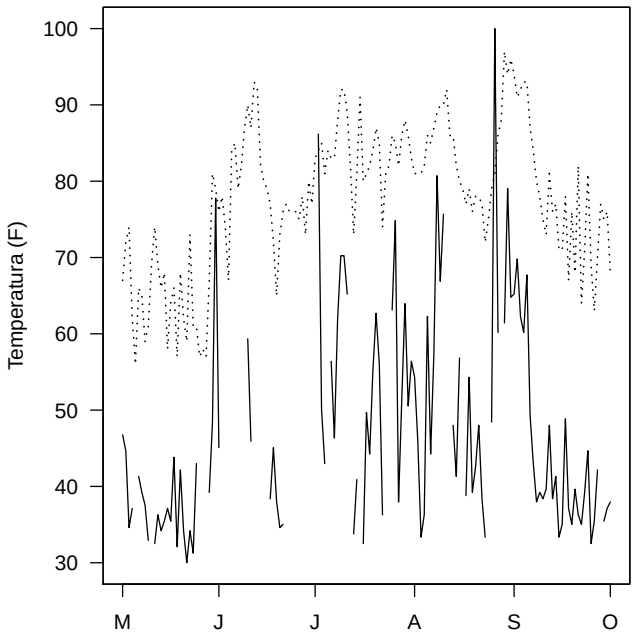


A próximo passo consiste em sobrepor sobre o mesmo gráfico o segundo conjunto de dados. A função par é usada novamente com o argumento new = TRUE, isso faz com que o próximo gráfico seja construido sobreposto ao gráfico antigo. Eixos e nomes dos eixos não devem ser mostrados, eles deverão ser adicionados pela função axis.

Argumento:

new - Argumento lógico para especificar ser próximo gráfico deve ser construido sem apagar o gráfico antigo.

```
plot(dados$Temp,
    type = "l",
    lty = 3,
    ylim = c(30, 100),
    lwd = 1.4,
    xaxt = "n",
    ylab = "Temperatura (F)",xlab = "")

axis(side = 1,
    at = c(1, cumsum(table(dados[, 5]))),
    label = c("M", "J", "J", "A", "S", "O"))

par(new = TRUE)

plot(dados$Ozone,
    type = "l",
    xlab = "", ylab = "",
    yaxt = "n", xaxt = "n",
    lty = 1)
```



Pode-se adicionar o eixo e nome do eixo na margem direita do gráfico. A função mtext é usada para adicionar texto nas margens do gráfico.

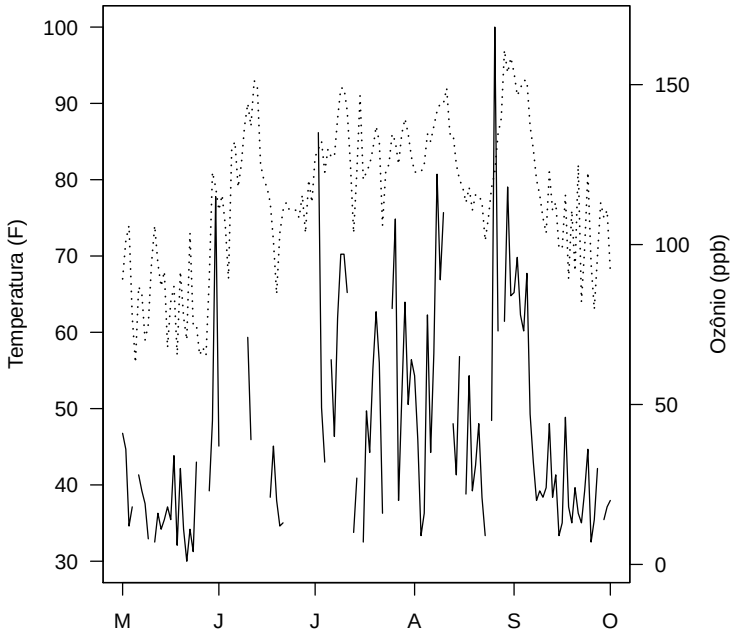
Argumentos:

side - Posição para o eixo e texto na função mtext.

line - Linha para mostrar texto. Valores próximo a 1 corresponde a posição próximo ao eixo.

```
plot(dados$Temp,
    type = "l",
    lty = 3,
    ylim = c(30, 100),
    lwd = 1.4,
    xaxt = "n",
    ylab = "Temperatura (F)",xlab = "")

axis(side = 1,
    at = c(1, cumsum(table(dados[, 5]))),
    label = c("M", "J", "J", "A", "S", "O"))

par(new = TRUE)

plot(dados$Ozone,
    type = "l",
    xlab = "", ylab = "",
    yaxt = "n", xaxt = "n",
    lty = 1)

axis(side = 4)

mtext(text = "Ozônio (ppb)",
    side = 4,
```

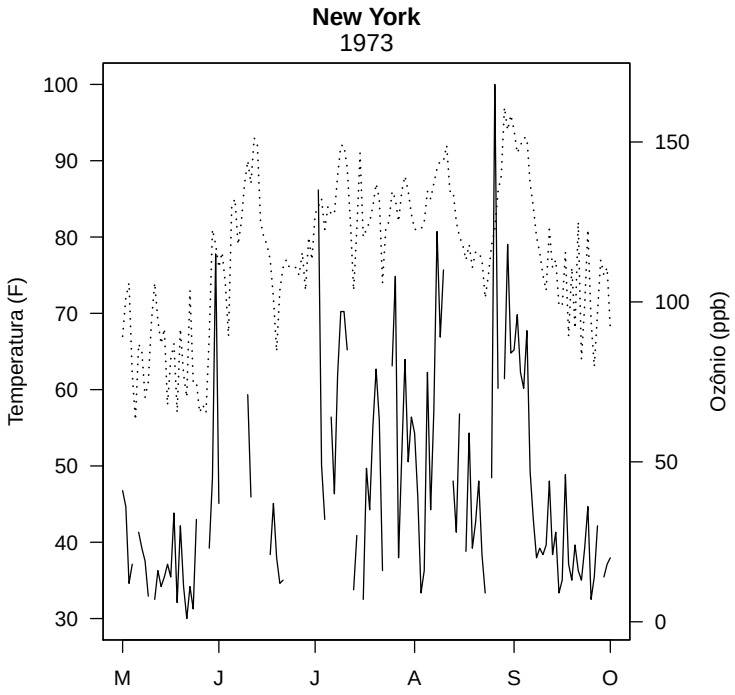
```
    las = 0,
    line = 3)
```



O título e subtítulo podem ser adicionados ao gráficos usando a função title.
Argumentos:
line - Linha para mostrar texto. Valores próximo a 1 corresponde a posição próximo ao eixo.
cex - Tamanho do texto.

```
plot(dados$Temp,
    type = "l",
    lty = 3,
    ylim = c(30, 100),
    lwd = 1.4,
    xaxt = "n",
    ylab = "Temperatura (F)",xlab = "")

axis(side = 1,
    at = c(1, cumsum(table(dados[, 5]))),
    label = c("M", "J", "J", "A", "S", "O"))

par(new = TRUE)

plot(dados$Ozone,
    type = "l",
    xlab = "", ylab = "",
    yaxt = "n", xaxt = "n",
    lty = 1)

axis(side = 4)
```

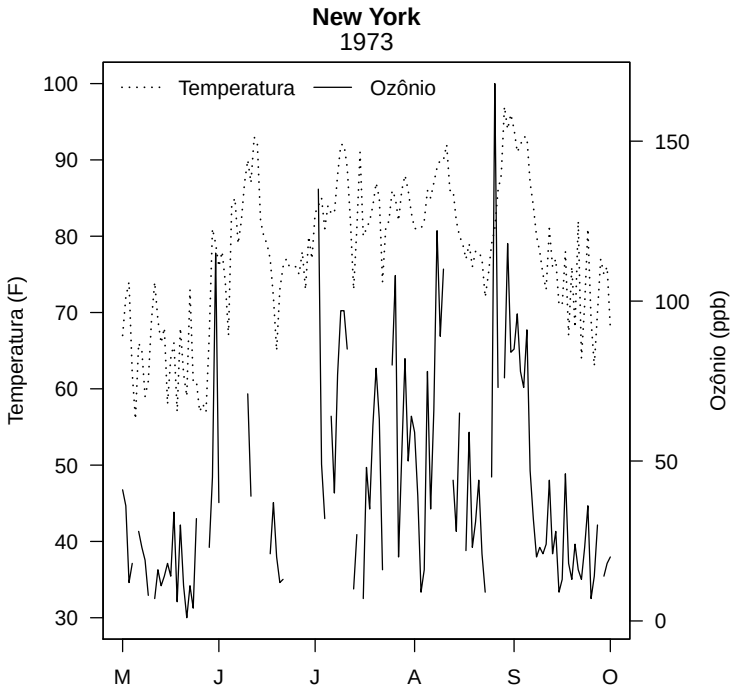
```
mtext(text = "Ozônio (ppb)",
    side = 4,
    las = 0,
    line = 3)

title(main = "New York",
    line = 1.5)

title(main = "1973",
    line = 0.5,
    cex = 0.8,
    font.main = 1)
```



Para finalzar pode-se adicionar legenda ao gráfico.
Argumentos:
legend - Especificar o texto para a legenda.
lty e **lwd** - Especificar tipo e espessura das linhas de cada nível.
bty - Especificar tipo de borda da legenda.
ncol - Especificar número de colunas para a legenda.

```
plot(dados$Temp,
    type = "l",
    lty = 3,
    ylim = c(30, 100),
    lwd = 1.4,
    xaxt = "n",
    ylab = "Temperatura (F)",xlab = "")

axis(side = 1,
    at = c(1, cumsum(table(dados[, 5]))),
```

```
    label = c("M", "J", "J", "A", "S", "O"))

par(new = TRUE)

plot(dados$Ozone,
    type = "l",
    xlab = "", ylab = "",
    yaxt = "n", xaxt = "n",
    lty = 1)

axis(side = 4)

mtext(text = "Ozônio (ppb)",
    side = 4,
    las = 0,
    line = 3)

title(main = "New York",
    line = 1.5)

title(main = "1973",
    line = 0.5,
    cex = 0.8,
    font.main = 1)

legend("topleft",
    legend = c("Temperatura", "Ozônio"),
    lty = c(3, 1),
    lwd = c(1.4, 1),
    bty = "n",
    ncol = 2)

par(resetPar)
```



3.10 Histogramas

Dados

Dados do tempo de espera entre erupções e a duração da erupção do gêiser Old Faithful em Yellowstone National Park, Wyoming, EUA.

```
data(faithful)
dados <- faithful
str(dados)

## 'data.frame': 272 obs. of  2 variables:
##  $ eruptions: num  3.6 1.8 3.33 2.28 4.53 ...
##  $ waiting  : num  79 54 74 62 85 55 88 85 51 85 ...
```

Plot

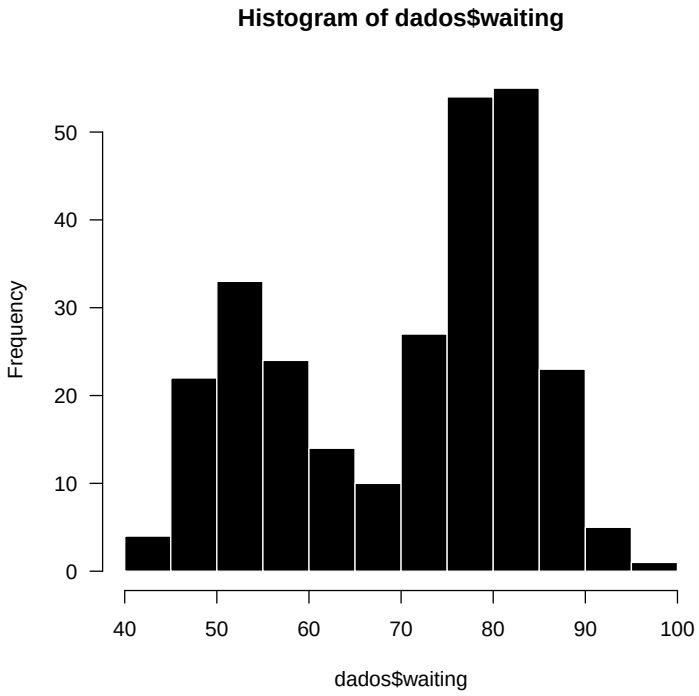
Para construir um histograma usamos a função hist.

Argumentos:

col - Cor para as barras.

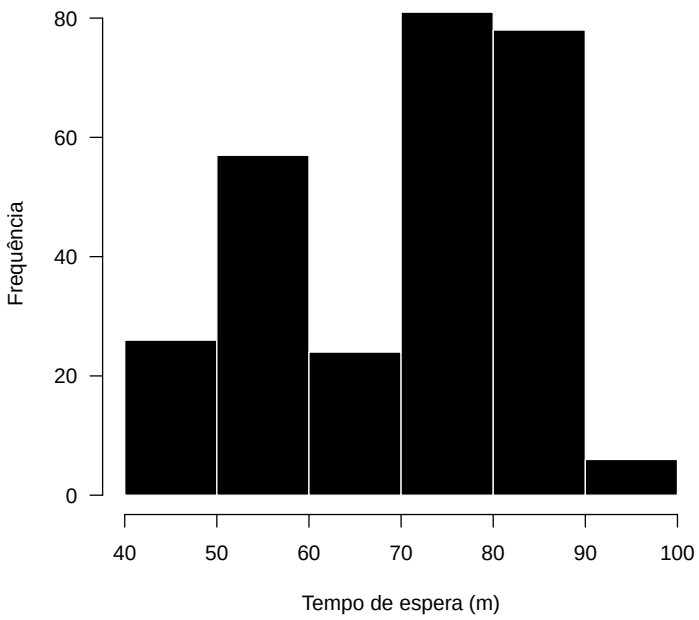
border - Cor para as bordas da barra.

```
hist(dados$waiting,
    las = 1,
    col = "black",
    border = "white")
```



Pode-se alterar o critério dos limites de cada barra do hostograma.
Argumento:
breaks - Especificar os limites de para o agrupamento dos valores em cada barra.

```
hist(dados$waiting,
    las = 1,
    col = "black",
    border = "white",
    breaks = c(40, 50, 60, 70, 80,90, 100),
    xlab = "Tempo de espera (m)", ylab = "Frequência",
    main = "")
```



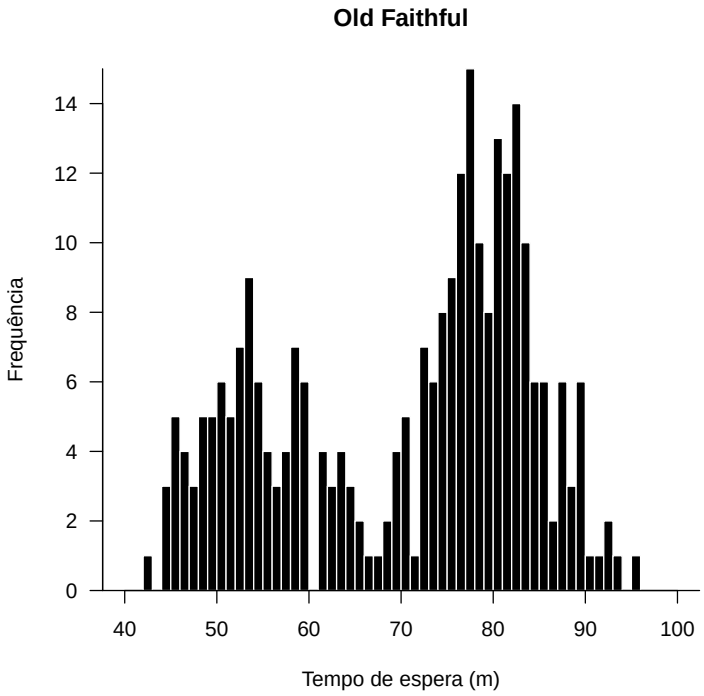
Pode-se ainda fazer com que os eixos fiquem juntos. Para isso é preciso definir o método de calcular os limites do eixo antes de fazer o histograma na função par. Além disso é preciso usar a função box para adicionar um contorno ao gráfico

Argumentos:

yaxs - Alterar o estilo de cálculo dos limites dos eixos.

bty - Altera o tipo de caixa (box) usado envolta do gráfico.

```
par(yaxs = "i")

hist(dados$waiting,
    las = 1,
    col = "black",
    border = "white",
    breaks = c(40:100),
    xlab = "Tempo de espera (m)",ylab = "Frequência",
    main = "Old Faithful")

box(bty = "l")

par(resetPar)
```



A função hist permite por também mostrar a densidade dos valores, não apenas a contagem em cada intervalo dos dados. Adicionalmente pode-se adicionar a linha de densidade sobre o histograma.

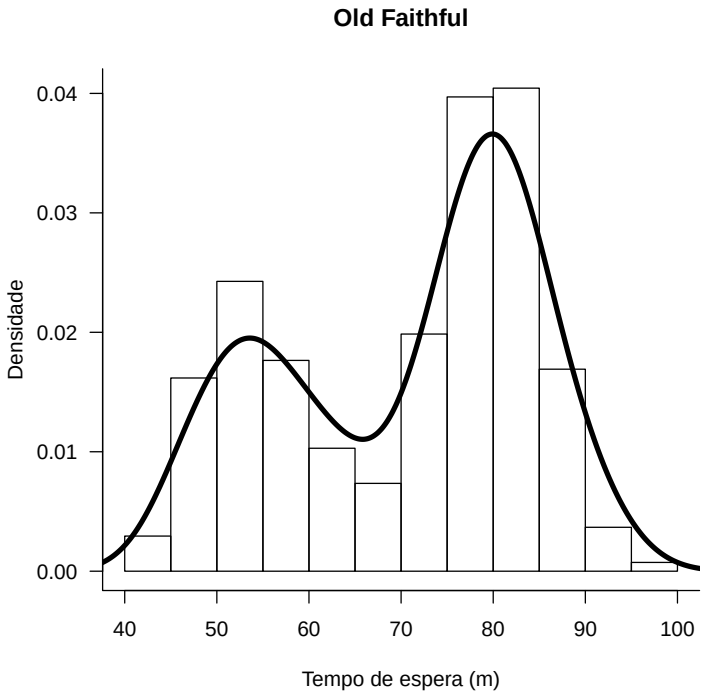
Argumentos:

prob - Argumento lógico para especificar se densidade deve ser mostrada.

lwd - Espessura da linha.

bty - Tipo de contorno para o gráfico.

```
hist(dados$waiting,
    prob = TRUE,
    las = 1,
    xlab = "Tempo de espera (m)", ylab = "Densidade",
    main = "Old Faithful")

lines(density(dados$waiting),
    lwd = 4)

box(bty = "l")
```



3.11 Painéis gráficos

Dados

Dados iris descrevem três espécies de iris com medidas do comprimento e largura de pétalas e sépalas.

```
data(iris)
dados <- iris[, 1:4]
str(dados)

## 'data.frame': 150 obs. of  4 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
```

Plot

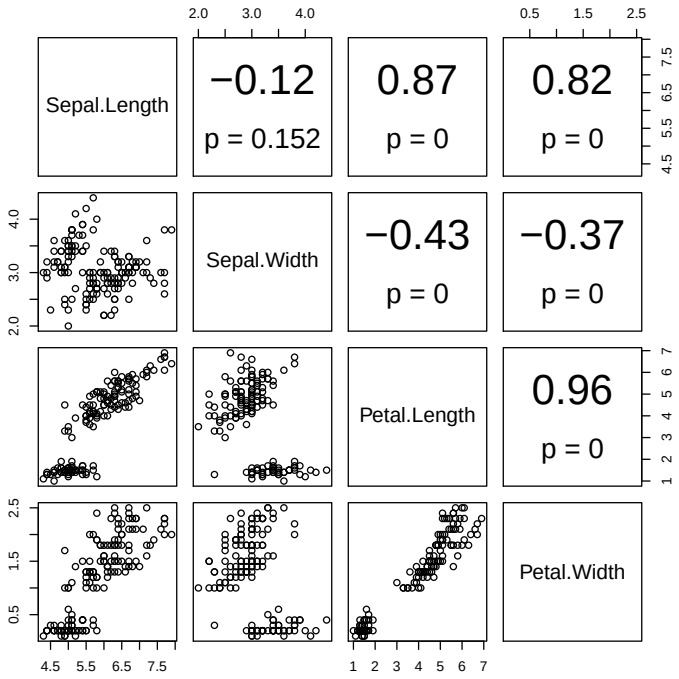
A função pairs permite visualizar relações entre várias variáveis simultaneamente. O painel resultante sera composto por tantas linhas e colunas de gráficos quanto váriaveis. Adicionalmente pose-se construir funções para adiconar elementos espécificos na diagoral ou painel inferior ou superior de maneira independente. Abaixo segue dois exemplos modificados de funções mostrada da ajuda da função pairs.

```
panel.cor <- function(x, y, ...) {
    par(usr = c(0, 1, 0, 1))
    res <- cor.test(x, y, ...)
    txt <- as.character(round(res$estimate, 2))
    txt2 <- paste("p =", round(res$p.value, 3))
    text(0.5, 0.7, txt, cex = 3)
    text(0.5, 0.25, txt2, cex = 2)
```

```
}
panel.hist <- function(x, ...) {
    usr <- par("usr")
    on.exit(par(usr))
    par(usr = c(usr[1:2], 0, 1.5))
    h <- hist(x, plot = FALSE)
    breaks <- h$breaks
    nB <- length(breaks)
    y <- h$counts
    y <- y/max(y)
    rect(breaks[-nB], 0, breaks[-1], y, col = "black", border = "white", ...)
}
```

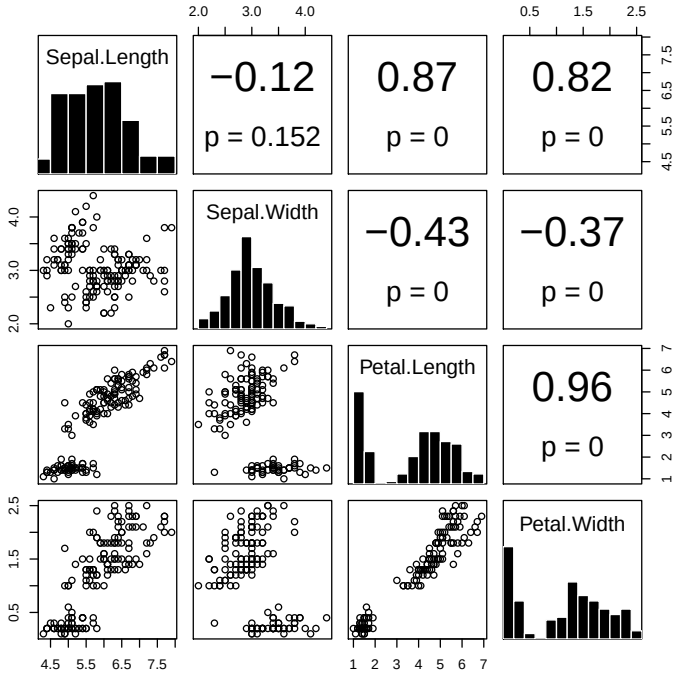
Argumento:
upper.panel - Função usada para os gráficos do painel supertior.

```
pairs(dados,
    upper.panel = panel.cor)
```



Argumento:
diag.panel - Função usada para os gráficos da diagonal do painel.

```
pairs(dados,
    upper.panel = panel.cor,
    diag.panel = panel.hist)
```

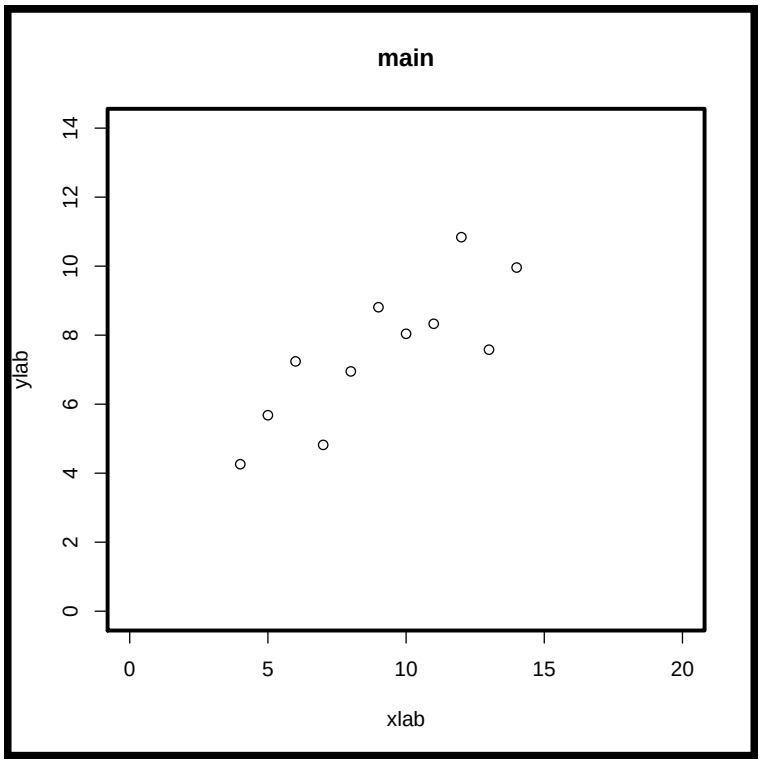


4 Múltiplos gráficos

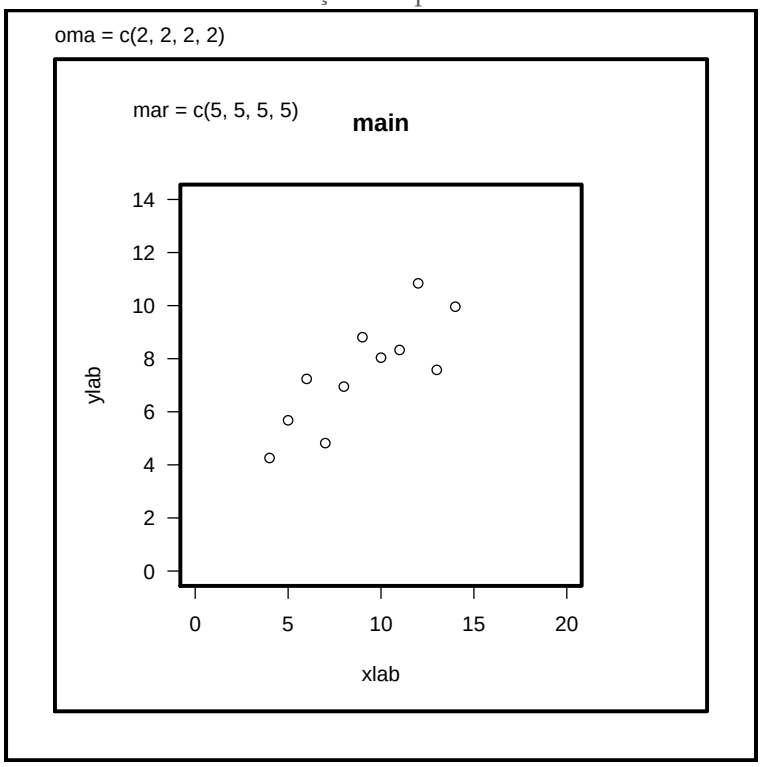
Um dos recurso disponíveis no R é a possibilidade de construir vários gráficos agrupados lado a lado. Estes podem ser exportados juntos em um único arquivo. É preciso entender o funcionamento das margens do dipositivo para personalizar o layout da dos gráficos múltiplos.

4.1 Margens

O argumento mar deve ser especificado antes de começar o construção do gráfico, na função par. O argumento mar é um vetor numérico que indica o número de linhas em cada uma das margens, segundo a forma c(margem inferior, margem à esquerda, margem superior, margem à direita). O padrão é mar = c(5.1, 4.1, 4.1, 2.1). O resultado desta configuração é que a margem esquerda é bem menor que os demais. Ele deve ser alterado para adicionar um eixo na margem direita por exemplo. Na figura abaixo a área do gráfico, onde encontram-se os pontos, é definida até os eixos. A margem da figura é definida entre os eixos e a borda exterior.



Adicionalmente é possivel definir o argumento oma na função par. Este argumento define o tamanho da margem externa a da figura. Da mesma forma que o argumento mar, o argumento oma é um vetor numérico que indica o número de linhas em cada uma das margens externas a figura, segundo a forma c(margem inferior, margem à esquerda, margem superior, margem à direita). O padrão de oma = c(0, 0, 0, 0). A figura abaixo mostra o resultado da alteração dos parâmetros mar e oma em um gráfico.

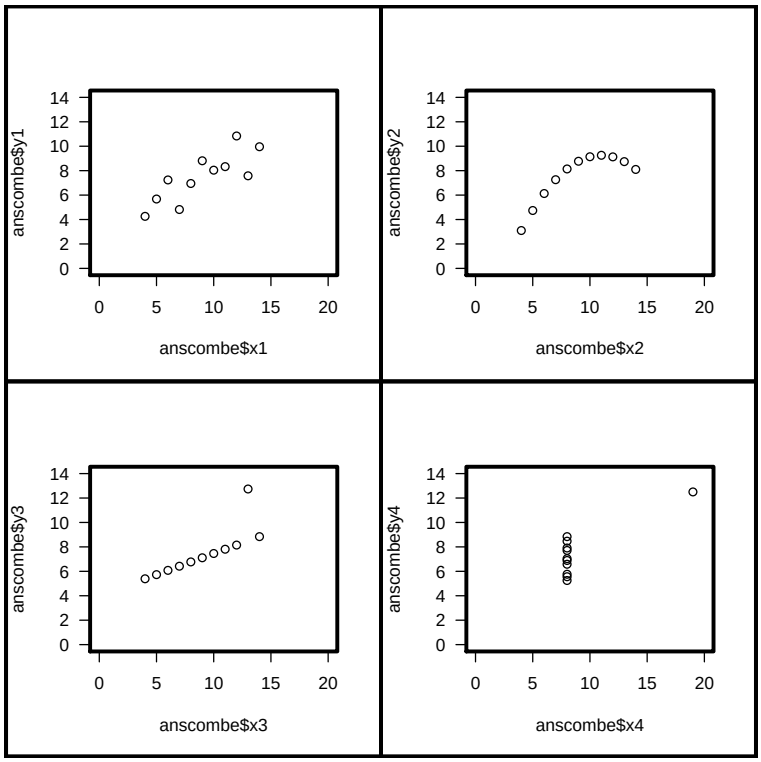


4.2 Gráficos multiplos

Uma maneira simples de construir vários gráficos em um mesmo dispositivo é usar os argumentos mfrow e mfcol na função par. Estes argumentos mfrow e mfcol devem ser especificadas antes de começar a construção dos gráficos, na função par. Os dois argumentos funcionam da mesma maneira, fornecendo um vector da forma de c(nr, nc), sendo que nr especifica o número de colunas e nc especifica o número de colunas de gráficos no dispositivo. Se fornecido o 4 e 2 resultará em oito gráficos no mesmo dispositivo. A diferença entre mfrow e mfcol é que o argumento mfrow os gráficos são construidos pelas linhas, da esquerda para a direita e depois passam para a segunda linha. Quando o argumento mfcol é usado os gráficos são construidos pelas colunas, antes a primeira coluna e depois a segunda coluna. Alternativamente pose-se usar as funções layout e split.screen pode produzir outros layout gráficos.

O exemplo abaixo ilustra o funcionamento da função mfrow sem alterar os argumentos mar e oma. As linhas da borda da figura e com os limites de cada gráfico foram destacadas para ilustrar o funcionamento das margens. Originalmente elas não são mostradas.

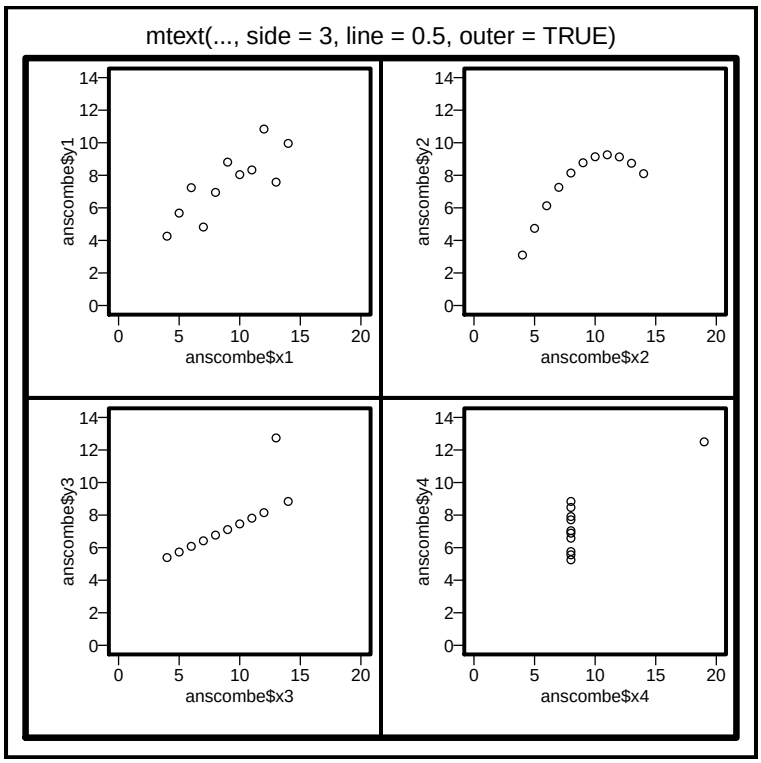
```
par(mfrow = c(2, 2), las = 1)

plot(anscombe$x1, anscombe$y1, xlim = c(0, 20), ylim = c(0, 14))
plot(anscombe$x2, anscombe$y2, xlim = c(0, 20), ylim = c(0, 14))
plot(anscombe$x3, anscombe$y3, xlim = c(0, 20), ylim = c(0, 14))
plot(anscombe$x4, anscombe$y4, xlim = c(0, 20), ylim = c(0, 14))

par(resetPar)  # Restaurar originais.
```



Na figura acima o espaço entre os gráficos poderia ser melhor aproveitado se as margens dos gráficos fossem alteradas. Abaxo segue o mesmo exemplo apenas alterando os argumentos mar e oma. Novamente as linhas que limitam os gráficos foram destacadas para visualização. A função mtext pode ser usada para adicionar um título geral para o conjunto de gráficos.

```
par(mfrow = c(2, 2), las = 1, mar = c(4, 4, 0.5, 0.5), oma = c(1, 1, 2.5, 1))
```

```r
plot(anscombe$x1, anscombe$y1, xlim = c(0, 20), ylim = c(0, 14), mgp = c(1.5, 0.5, 0))
plot(anscombe$x2, anscombe$y2, xlim = c(0, 20), ylim = c(0, 14), mgp = c(1.5, 0.5, 0))
plot(anscombe$x3, anscombe$y3, xlim = c(0, 20), ylim = c(0, 14), mgp = c(1.5, 0.5, 0))
plot(anscombe$x4, anscombe$y4, xlim = c(0, 20), ylim = c(0, 14), mgp = c(1.5, 0.5, 0))

par(resetPar)  # Restaurar originais.
```



5 Exportar gráficos

Até agora usamos os dispositivos gráficos para visualizar os gráficos, para exportar um gráfico para um arquivo vamos usar outros dispositivos. Os dispositivos para exportar são em dois grupos. Arquivos de imagens que são dependentes de resuluçoes como jpeg(), bmp(), tiff() e png(). O outro grupo são arquivos vectorias independentes de resuluçoes como pdf(), svg(), e postscript().

A estrutura básica de expostação é a seguinte:

```r
pdf("Nome do arquivo.pdf")

# Comandos gráficos

dev.off()
```

A função dev.off() é usada para finalizar o dispositivo de exportação. Se não for finalizada ela exporta cada gráfico construido no mesmo arquivo.

Para exportar um exemplo em cada grupo de arquivo de imagem. Pode-se utilizar o conjunto de dados anscombe.

```r
data(anscombe)
dados <- anscombe
str(dados)
```

```
## 'data.frame': 11 obs. of  8 variables:
##  $ x1: num  10 8 13 9 11 14 6 4 12 7 ...
##  $ x2: num  10 8 13 9 11 14 6 4 12 7 ...
##  $ x3: num  10 8 13 9 11 14 6 4 12 7 ...
##  $ x4: num  8 8 8 8 8 8 8 19 8 8 ...
##  $ y1: num  8.04 6.95 7.58 8.81 8.33 ...
##  $ y2: num  9.14 8.14 8.74 8.77 9.26 8.1 6.13 3.1 9.13 7.26 ...
##  $ y3: num  7.46 6.77 12.74 7.11 7.81 ...
##  $ y4: num  6.58 5.76 7.71 8.84 8.47 7.04 5.25 12.5 5.56 7.91 ...
```

5.1 Gráfico como imagem

No caso dos arquivos imagens pode-se exportar em usando a função png. É preciso expecificar as dimensão da figura que será exportada bem como a resulução da imagem.

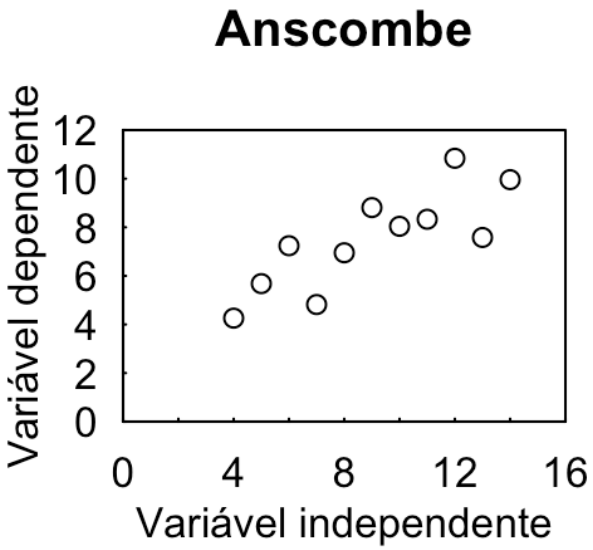
Argumentos:

res - Resulução em ppi.

height - Altura em pixels.

width - Largura em pixels.

```
png("Anscombe.png", res = 200, height = 600, width = 600)
plot(dados$x1, dados$y1,
    xlab = "Variável independente", ylab = "Variável dependente",
    main = "Anscombe",
    las = 1,
    xlim = c(0, 16), ylim = c(0, 12),
    xaxs = "i", yaxs = "i",
    xaxp = c(0, 16, 8), yaxp = c(0, 12, 6),
    tck = 0.01,
    mgp = c(1.5, 0.5, 0))
dev.off()

## null device
##   1
```



A imagem exportada perde a formatação e o layout. Para reajustar é preciso alterar vários parâmertros gráficos, como por exemplo, mex, cex, cex.main, cex.lab e cex.axis.

5.2 Gráfico vectorial

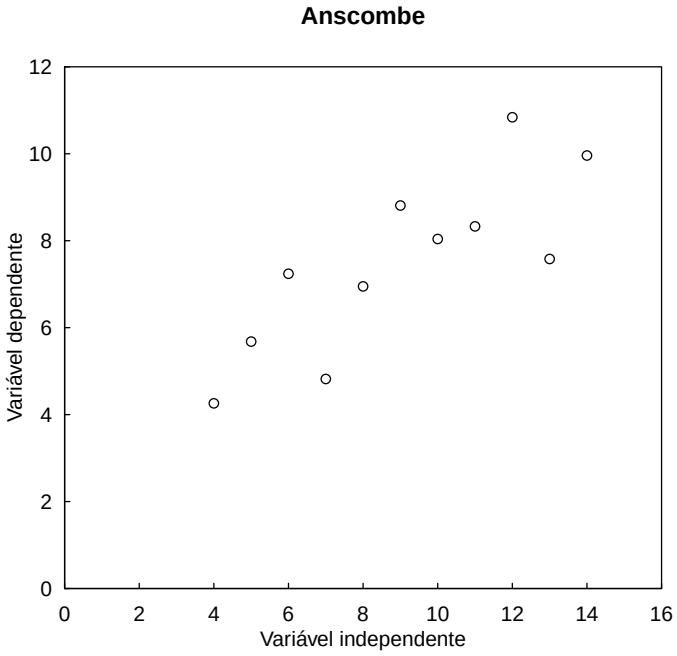
Para exportar em arquivo vectorial a resolução não precisa ser ajustada, apenas as dimensões do arquivo de saída.

Argumentos:

height - Altura em polegadas.

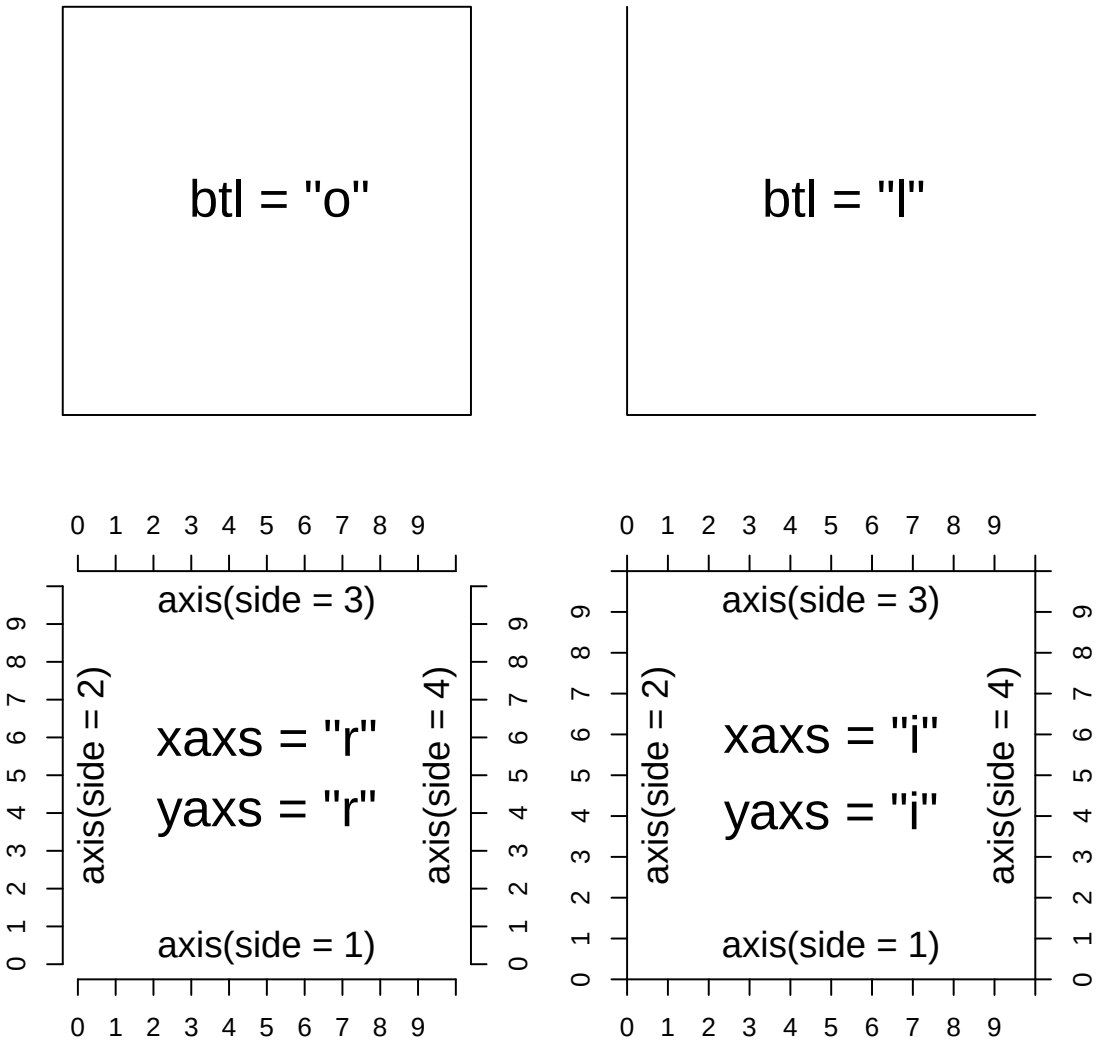
width - Largura em polegadas.

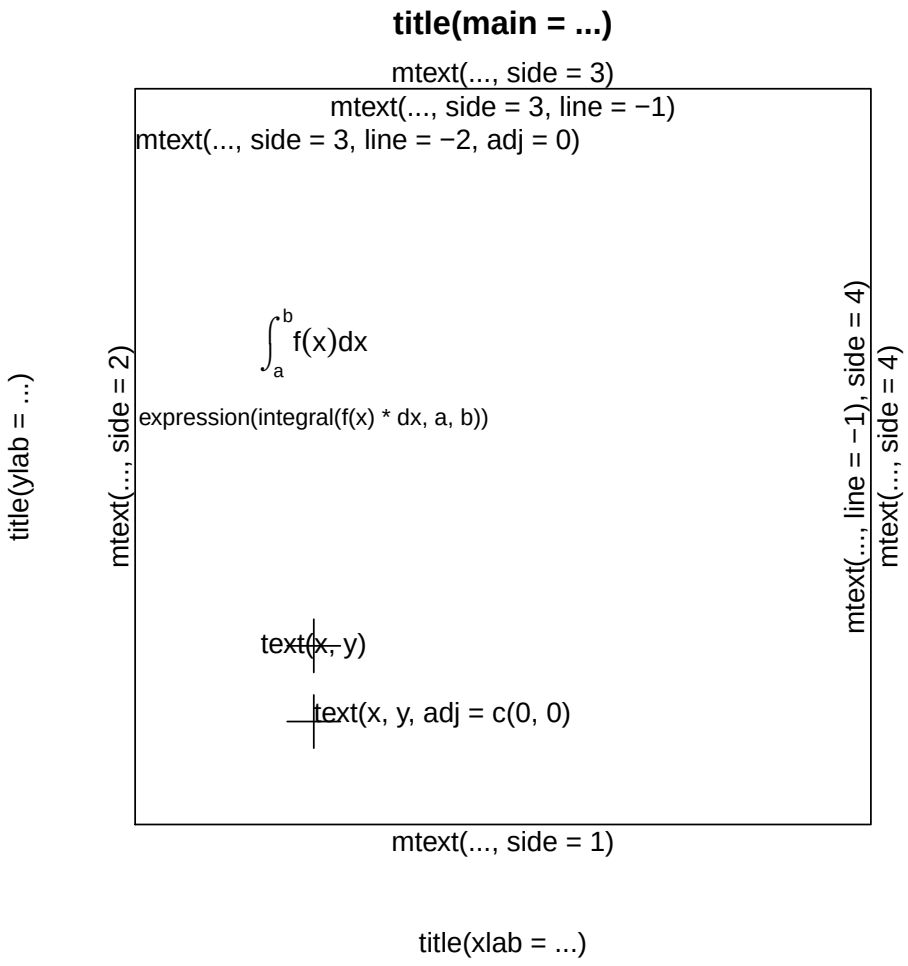
```
pdf("Anscombe.pdf", width = 6, height = 6)
plot(dados$x1, dados$y1,
    xlab = "Variável independente", ylab = "Variável dependente",
    main = "Anscombe",
    las = 1,
    xlim = c(0, 16), ylim = c(0, 12),
    xaxs = "i", yaxs = "i",
    xaxp = c(0, 16, 8),yaxp = c(0, 12, 6),
    tck = 0.01,
    mgp = c(1.5, 0.5, 0))
dev.off()

## pdf
##   2
```

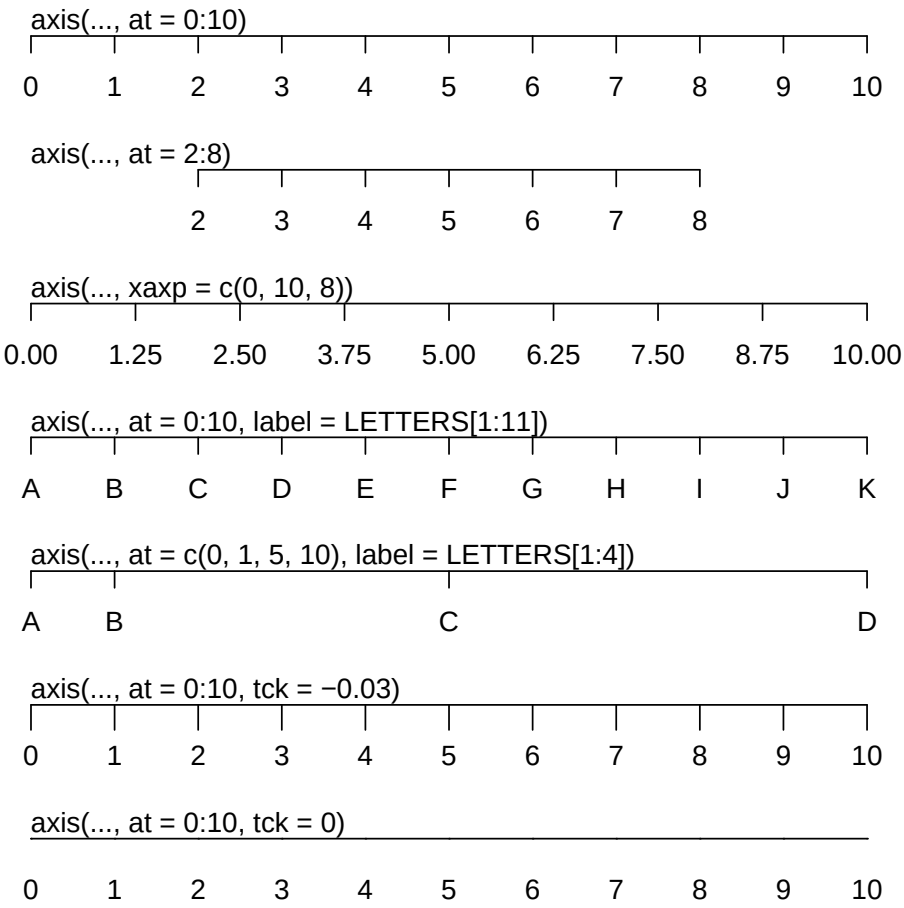


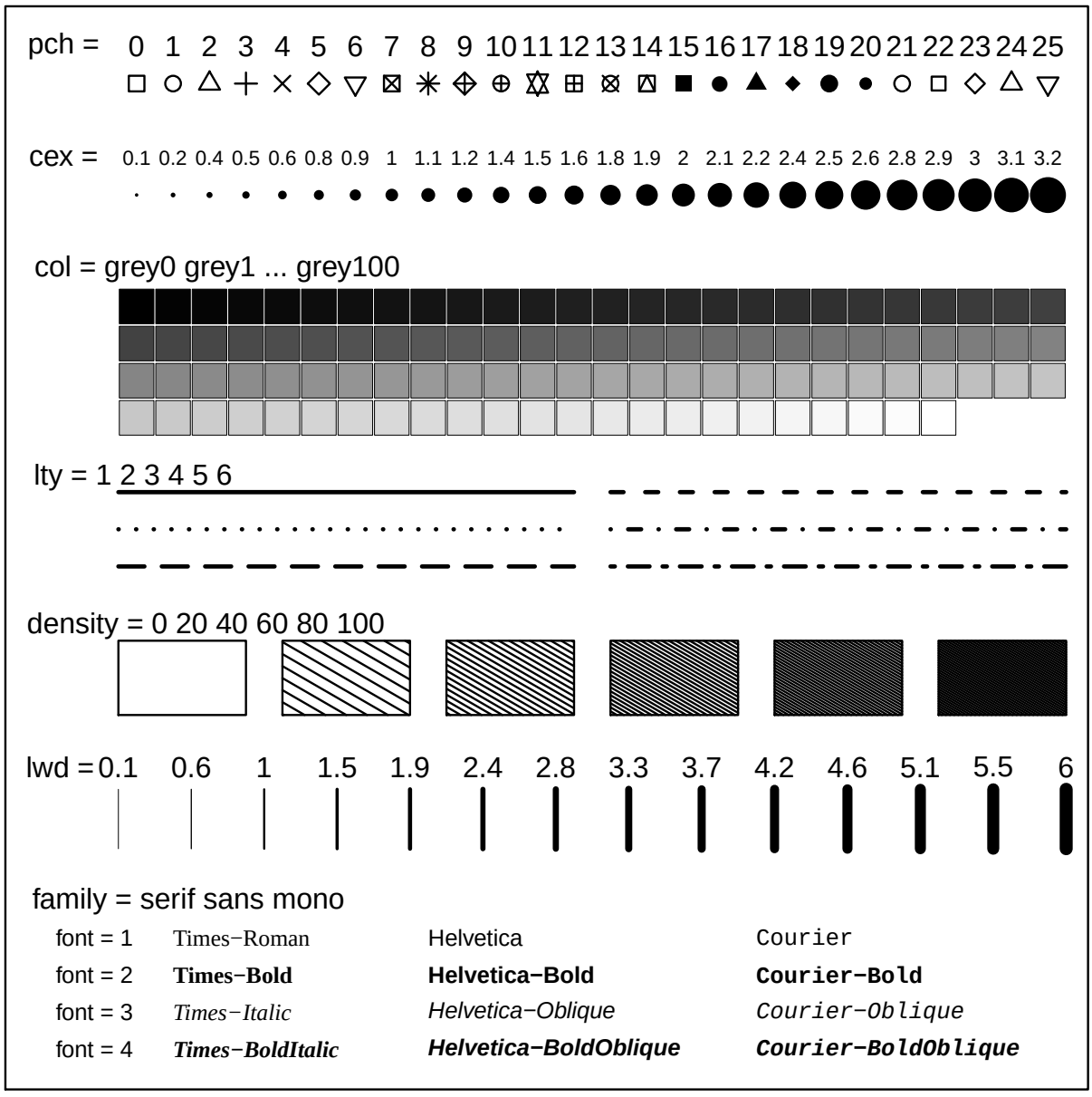
No caso do arquivo vectorial o resultado é mais próximo ao visualizado dispositivo de visualização gráfica. Também é possivel ajustar os parâmetros gráficos para um melhor resultado. A exportação no formato vectorial vectorial é recomendara, por ser mais fácil e por não perder a resolução ao aplicar zoom.

6 Esquema de ajuda gráfica




title(main = ...)
mtext(..., side = 3)
mtext(..., side = 3, line = −1)
mtext(..., side = 3, line = −2, adj = 0)
$\int_a^b f(x)dx$
expression(integral(f(x) * dx, a, b))
text(x, y)
text(x, y, adj = c(0, 0)
mtext(..., side = 1)
title(xlab = ...)
title(ylab = ...)
mtext(..., side = 2)
mtext(..., line = −1), side = 4)
mtext(..., side = 4)


axis(..., at = 0:10)
0 1 2 3 4 5 6 7 8 9 10
axis(..., at = 2:8)
2 3 4 5 6 7 8
axis(..., xaxp = c(0, 10, 8))
0.00 1.25 2.50 3.75 5.00 6.25 7.50 8.75 10.00
axis(..., at = 0:10, label = LETTERS[1:11])
A B C D E F G H I J K
axis(..., at = c(0, 1, 5, 10), label = LETTERS[1:4])
A B C D
axis(..., at = 0:10, tck = −0.03)
0 1 2 3 4 5 6 7 8 9 10
axis(..., at = 0:10, tck = 0)
0 1 2 3 4 5 6 7 8 9 10


pch = 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
cex = 0.1 0.2 0.4 0.5 0.6 0.8 0.9 1 1.1 1.2 1.4 1.5 1.6 1.8 1.9 2 2.1 2.2 2.4 2.5 2.6 2.8 2.9 3 3.1 3.2
col = grey0 grey1 ... grey100
lty = 1 2 3 4 5 6
density = 0 20 40 60 80 100
lwd = 0.1 0.6 1 1.5 1.9 2.4 2.8 3.3 3.7 4.2 4.6 5.1 5.5 6
family = serif sans mono
font = 1 Times−Roman Helvetica Courier
font = 2 Times−Bold Helvetica−Bold Courier−Bold
font = 3 Times−Italic Helvetica−Oblique Courier−Oblique
font = 4 Times−BoldItalic Helvetica−BoldOblique Courier−BoldOblique